▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 14

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

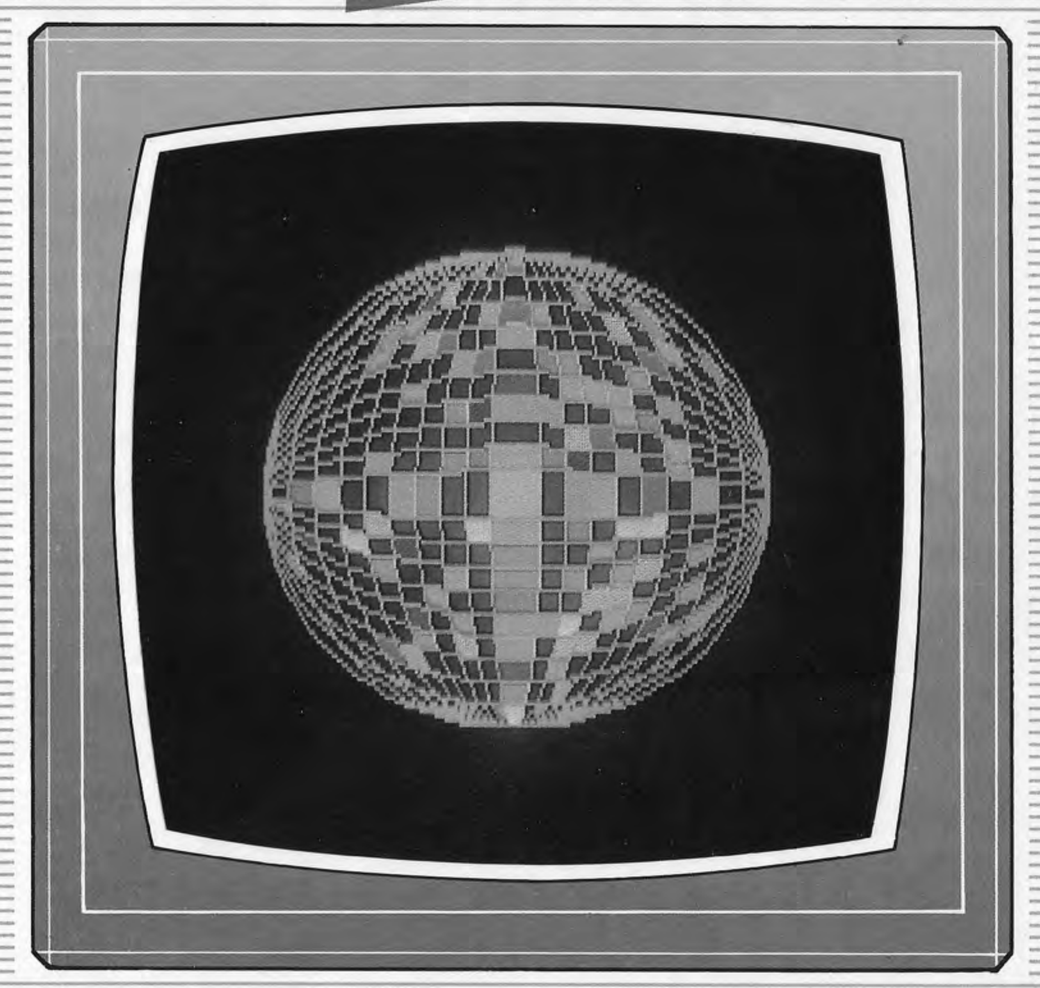
▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 14

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼

▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.

Diseño y maquetación:
BRAVO-LOFISH.

Fotografía:
EQUIPO GALATA.

Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de AULA DE INFORMATICA APLICADA (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomuniación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanero, Licenciada en Informática. APLICACIONES: Soledad Tamariz, Diplomada en Telecomunicación. OTROS LENGUAJES (sistemas operativos): Domingo Villaseñor, Diplomado en Informática, y Lenguaje C: Enrique Serrano, Ingeniero en Telecomunicación.

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-106-1
ISBN de la obra: 84-7688-068-7

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M-5-677-1987
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 810 52 13. 28020 Madrid.

Junio, 1987.
P.V.P. Canarias: 335,–.

INDICE



INFORMATICA BASICA

INSTRUCCIONES Y PROGRAMAS



Instrucciones. Tipos

BASICAMENTE las instrucciones son conjuntos de caracteres que representan órdenes de un lenguaje de programación y que se da a un ordenador para que ejecute una o varias funciones u operaciones.

Todas las instrucciones contienen dos partes fundamentales:

— **Código de operación.** Indica a la Unidad de Control la operación que tiene que realizar.

— **Dirección de los operandos.** Indica dónde están situados los datos que tiene que manejar para ejecutar la instrucción deseada.

En cuanto al número de instrucciones que puede realizar un ordenador, varía de unos a otros. Esta colección de operaciones se denomina **juego o repertorio de instrucciones.**

El número de instrucciones suele oscilar entre unas 20 para ordenadores muy sencillos y unas 200 para los más potentes, aunque lo normal son 100.

Los distintos tipos de instrucciones son los siguientes:

— **Instrucciones de tres direcciones.**

CODIGO DE OPERACION	DIRECCION DEL PRIMER OPERANDO	DIRECCION DEL SEGUNDO OPERANDO	DIRECCION DEL RESULTADO

Instrucción de tres direcciones.

Como vemos en la figura, los campos intermedios indican las direcciones de memoria donde se encuentran almacenados los operandos, y en el tercer campo se encuentra la dirección de dónde se almacenará el resultado de aplicar la operación indicada por el código de operación sobre ambos operandos.

— **Instrucciones de dos direcciones.**

CODIGO DE OPERACION	DIRECCION DEL PRIMER OPERANDO	DIRECCION DEL SEGUNDO OPERANDO Y DEL RESULTADO



Instrucción de dos direcciones.

En este caso sólo se indica la dirección del segundo y el tercer operando, ya que el resultado de operar sobre ellos se almacenará en la misma dirección donde se encontraba el segundo; es decir, una vez realizada la operación no podremos recuperar el valor del segundo operando, ya que habrá quedado «machacado» al introducir el resultado.

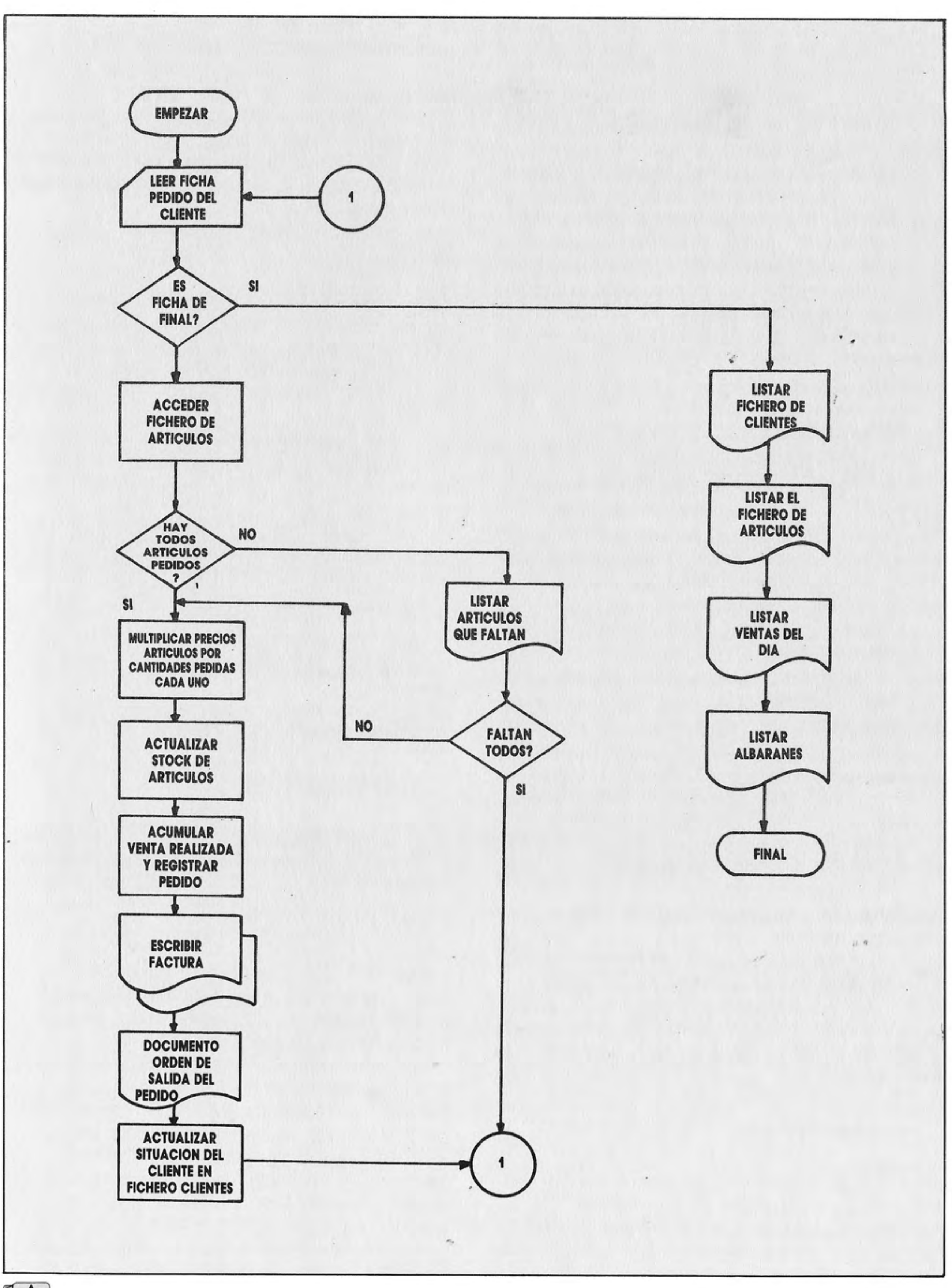
— **Instrucciones de una dirección.**

CODIGO DE OPERACION	DIRECCION DEL PRIMER OPERANDO



Instrucción de una dirección.

En los ordenadores que utilizan instrucciones con este formato se recurre a la utilización de registros internos en la máquina que contendrán el valor del segundo operando; de entre todos estos registros el más importante es el llamado **acumulador** (registro especial de la Unidad Aritmético-Lógica).



Organigrama simplificado de facturación.

— También existen las **instrucciones de cero operandos,** que no suelen utilizarse para operaciones aritméticas.

Podemos realizar otra clasificación de las instrucciones de acuerdo a la operación que realizan:

— **Aritméticas.** Suelen realizar las operaciones de suma y resta de operandos, aunque también hay veces que incluso pueden multiplicar y dividir.

— **De traslación.** Se utilizan para realizar transferencias de información, es decir, pueden mover datos de un registro a memoria, de memoria al acumulador, etcétera.

— **De bifurcación.** Sirven para romper la secuencia del programa. Dentro de este grupo podemos distinguir:

— **Condicionales.** Sólo efectúa la bifurcación si se cumple una cierta condición.

— **Incondicionales.** Se efectúa el salto a una instrucción almacenada en la memoria sin dependencia de los hechos ocurridos en el programa.

— **De interrupción.** Sirven para que el programa deje de ejecutarse. Por ejemplo, la instrucción de fin de programa.

— **De entrada o salida.** Se encargan del trasiego de los datos desde el ordenador hacia los dispositivos periféricos o viceversa.




Clasificación de las instrucciones por el tipo de operación.

Algoritmos

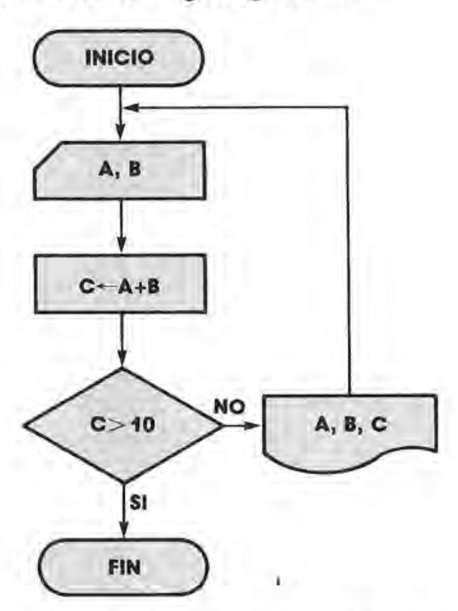
Se llama algoritmo al conjunto de pasos necesarios para la resolución de un problema. A veces no se ve muy clara la diferencia entre algoritmo y programa, ya que ambos son descripciones detalladas de los pasos para solucionar un problema; sin embargo, en un caso (algoritmo) el ordenador no puede ejecutar directamente estos pasos; en el caso del programa sí son ejecutables por el ordenador.

Los algoritmos se pueden representar de tres formas distintas:

— Mediante la descripción literal.

— Detallando cada uno de los pasos para resolver el problema.

— Mediante organigramas.




Ejemplo de un programa seguido por organigrama.

Organigramas

Un organigrama es la representación gráfica de la secuencia de un programa, o de la resolución de un problema. Se utilizan no sólo en programación, sino también en cualquier tarea de la vida cotidiana.

Hay dos tipos de organigramas:

— **Estáticos u organizacionales.** Se suelen utilizar para representar la estructura jerárquica de una empresa.

— **Dinámicos.** Son representaciones gráficas del sistema del proceso de información y facilitan la labor de análisis. Dentro de este grupo se encuentran:

— Los **organigramas de sistemas.** En los que se indican las entradas y salidas de información con sus soportes y archivos.

— Los **ordinogramas u organigramas de programas.** Representan con detalle los pasos necesarios para realizar el trabajo de programación.

Programas

Un programa se puede definir como un conjunto de sentencias (incluyendo las de definición de datos y ejecución) que se ejecutan por un ordenador para resolver el problema.

	1	2	3	4
CLIENTE ESPECIAL	SI	NO	NO	NO
I.T. COMPRAS 10.000	—	SI	NO	NO
10.000 I.T. COMPRAS	—	—	SI	NO
I.T. COMPRAS 100.000	—	—	—	SI
DESCUENTO 5 %	—	—	X	—
DESCUENTO 10 %	—	—	—	X
DESCUENTO 15 %	X	—	—	—
NO DESCONTAR	—	X	—	—


Tabla de decisión. Otra técnica de ayuda a la programación.



Estructura de los programas

Realmente los programas se ejecutan en el ordenador de forma secuencial, unas instrucciones detrás de otras; sin embargo, hay veces en que no deseamos que la instrucción que viene a continuación sea la siguiente en ejecutarse. Para ello, como vimos antes, existen los llamados «saltos», que son instrucciones que hacen que se rompa la secuencia del programa, yendo a ejecutar instrucciones que se encuentran en otro lugar del programa. Los saltos pueden ser condicionales o incondicionales.

Los bucles son estructuras del programa que hacen que las instrucciones encerradas en ellos se ejecuten un número repetido de veces. El caso más interesante es cuando el bucle se controla mediante un índice, el cual se modifica, en cada pasada, en una cantidad constante.

Una tercera estructura de programa consiste en la técnica de **decisiones múltiples.** Simboliza la posibilidad de elección entre varios caminos.



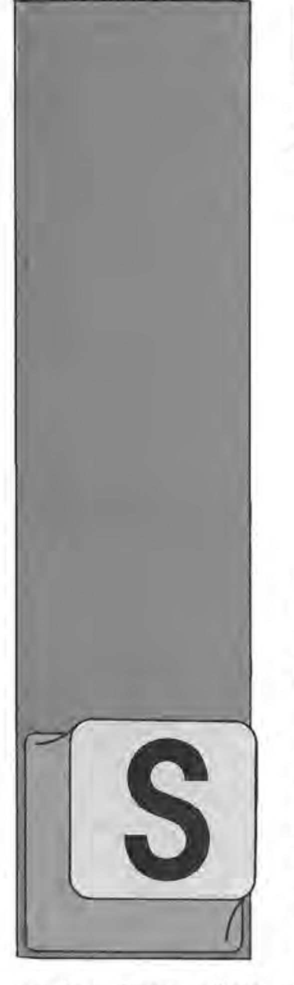
Técnicas y documentación de los programas

Una vez que el problema ha sido resuelto por parte de los analistas, empiezan a aparecer conceptos como: ficheros, cantidad, organizaciones, etc. Todo este conjunto de ideas informáticas queda plasmado en lo que se denominan **cuadernos de carga,** que contienen la descripción detallada de toda la aplicación, y mediante los cuales los programadores realizan su tarea.

La descripción de lo que es en sí cada programa se puede realizar mediante lenguaje natural, a modo de comentarios, o utilizando diagramas de bloques, tablas de decisión u organigramas.

MAQUINA 6502

COMMODORE 64



Comandos de almacenamiento

SON los comandos opuestos a los comandos de carga, es decir, se trata de almacenar un número en la memoria desde uno de los registros de trabajo del procesador. Presentan el siguiente formato:

STA
STX
STY

Están disponibles los mismos modos de direccionamiento, con excepción del direccionamiento inmediato, ya que siempre necesitamos una posición de memoria donde almacenar el contenido del registro con el que estemos trabajando. Su equivalente en BASIC es la instrucción POKE.

Una cuestión importante y diferenciadora de los comandos de almacenamiento sobre los de carga es que con los primeros, que son los que estamos estudiando ahora, no se alteran los flags del registro de estado del procesador, ya que lo único que se hace es transferir el valor desde el registro correspondiente a una posición de memoria. ¡Pero el registro no se altera!

A continuación se exponen los códigos de los comandos según los modos de direccionamiento utilizados.

Modos de direccionamiento	STA	STX	STY	Ejemplo
Absoluto	$80	$8E	$8C	STA$C000
Zeropage	$85	$86	$84	STX$FC
Absoluto indexado por X	$90	—	—	STA$C000,X
Absoluto indexado por Y	$99	—	—	STA$C000,Y
Zeropage indexado por X	$95	—	$94	STY$FC,X
Zeropage indexado por Y	—	$96	—	STX$FC,Y
Indirecto indexado en Y	$91	—	—	STA($BB),Y
Indirecto indexado en X	$81	—	—	STA($BB, X)

Bueno, creo que ha llegado el momento de hacer algo en código máquina y así tomar un respiro en cuanto a teoría.

Será algo muy sencillo y que puede que no sea de gran utilidad, pero al menos servirá para ilustrar lo que hemos visto hasta ahora.

Vamos a intentar que aparezca en pantalla el letrero «COMMODORE 64».

Para ello, lo primero que hacemos es cargar, por ejemplo, el ACU con el número correspondiente a la C, que, como puede verse en cualquier manual que presente un listado de los códigos de pantalla y códigos ASCII, tiene el valor 67=$43.

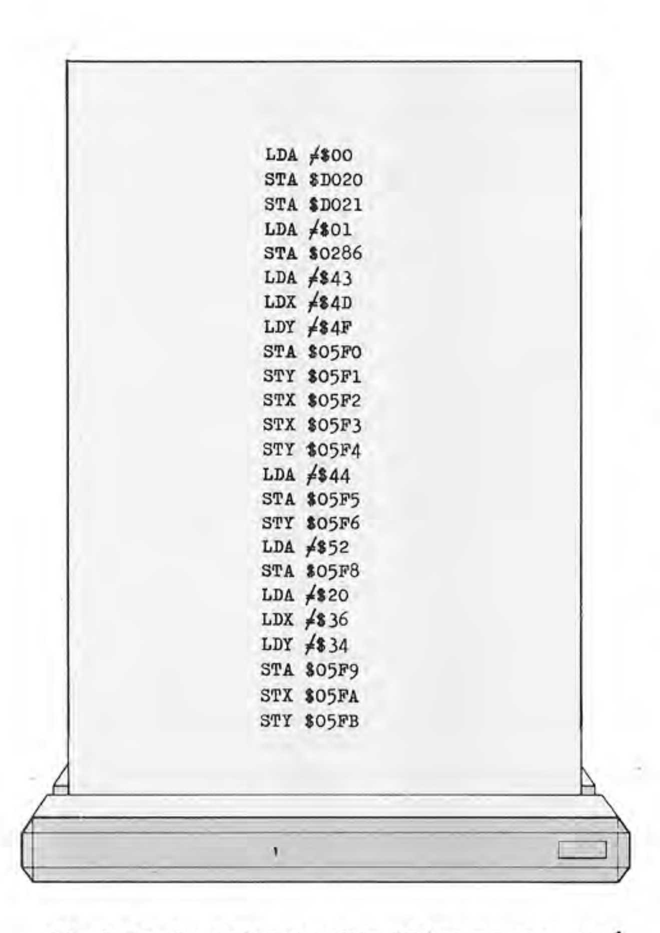
A continuación almacenamos dicha letra en una posición de la memoria que esté dentro de la RAM de vídeo (1024-2023).

Este proceso lo repetimos para todas las letras y habremos terminado.

Si además quiere que el borde, fondo y dibujo, tengan un determinado color, basta con que altere el valor de las posiciones de memoria:

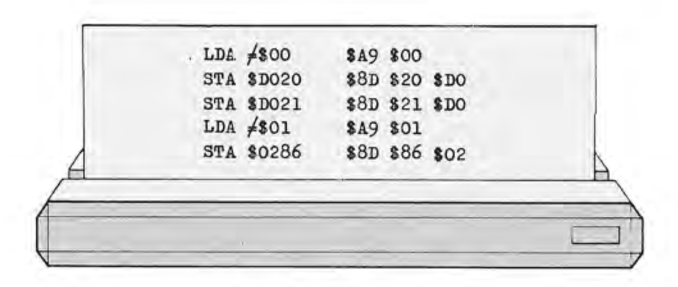
53280 = $D020 borde
53281 = $D021 fondo
646 = $0286 dibujo

Al final debe quedar algo así:



No intente introducirlo tal y como está, pues no funcionaría con ese formato.

Vamos a desglosar tan sólo la primera parte que pone borde y fondo de color negro, y dibujo de color blanco. El resto del programa, aunque es correcto, no se haría en la práctica así, ya que, como puede verse, todos los STA son posiciones consecutivas de memoria, y sería mucho más fácil y corto hacerlo mediante un bucle. Por tanto, lo dejaremos para cuando veamos los comandos de incremento y bifurcación.



Si tomamos los códigos de comando, nos quedaría la siguiente serie de números en hexadecimal:

$A9, $00, $8D, $20, $D0, $8D, $21, $D0, $A9, $01, $8D, $86, $02

Haciendo la transformación a decimal se obtiene:

169, 0, 141, 32, 208, 141, 33, 208, 169, 1, 141, 134, 2

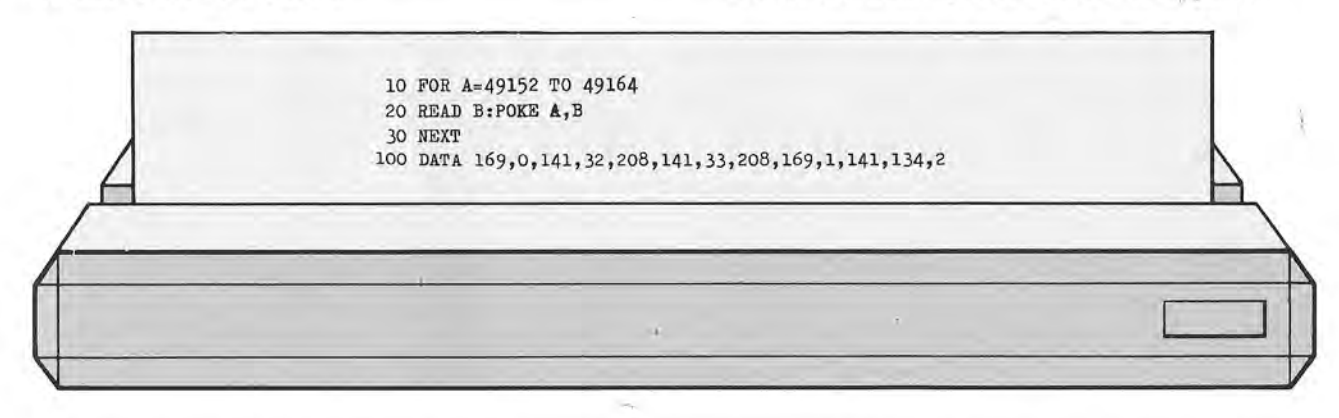
Ahora ya sólo queda introducir estos datos consecutivamente a partir de una determinada posición de memoria, y llamar a la rutina con SYS.

¿Dónde almacenar nuestra rutina en código máquina?

Hay varias zonas en las que podríamos hacerlo, pero de momento vamos a restringirnos al área comprendida entre 49152 = $C000 y 53247 = $CFFF, la cual representa 4 Kbytes, más que suficientes para un programa en lenguaje máquina.

He de decir que esta zona es área BASIC; por tanto , su rutina permanecerá ahí aunque borre su programa BASIC con NEW, ya que este comando, salvo que hayamos alterado ciertas variables en la página cero, «borrará» sólo desde la posición 2049=$0801 hasta 40959=$9FFF.

Bueno, pues «guardemos» la rutina.



Teclee este programa y ejecútelo con RUN. A continuación teclee en modo directo SYS 49152 y verá cómo la pantalla se vuelve negra y el cursor blanco. Tam-

bién puede incluir una línea 40 SYS 49152, y al teclear RUN producirá el mismo efecto.

Una aclaración antes de terminar:

—¿Qué ocurre cuando el procesador llega a la posición 49162? Pues que carga el contenido de la siguiente posición de memoria y la interpreta como un código de comando, ya que nadie le ha dicho que pare, que ya se ha terminado la rutina. Para comunicárselo debemos introducir el comando RTS al final de toda rutina en lenguaje máquina, que es equivalente al comando RETURN en BASIC y cuyo código es 96 = $60.

Así, pues, añadimos un «dato» más que es el número 96, modificamos el número 49164 a 49165 en la línea 10 y todo arreglado.



Ensambladores y monitores

Evidentemente no es necesario hacer todo el trabajo de buscar los códigos de los comandos con los que trabajamos, transferirlos al sistema decimal y crear un programa BASIC que los introduzca en memoria mediante sentencias READ y POKE. Para ello existen programas que aceptan los comandos que estamos aprendiendo y que se encargan de transformarlos para que puedan ser interpretados por el procesador. El programa que necesitamos es un «Assembler» y el lenguaje que nos ocupa es «lenguaje ensamblador».

Un «Assembler» suele ser una parte de un programa más grande llamado «Monitor» que consta de:

— Assembler.

— Disassembler.

— Monitor.

El Assembler ya hemos visto que sirve para que el ordenador entienda los comandos en lenguaje ensamblador.

El Disassembler es el programa opuesto al anterior. Lee un programa que está en memoria y lo visualiza en la forma que estamos viendo (LDA, STA...).

De la misma manera se pueden desensamblar zonas de memoria que no sean necesariamente programas, como la página cero, el sistema operativo o el intérprete BASIC. Es lo que suele decirse «ver las tripas del ordenador».

El monitor propiamente dicho puede controlar los registros del procesador de manera directa, así como modificar la memoria, arrancar programas en Assembler y efectuar las operaciones de grabación y carga desde los periféricos.

Además, suele tener una serie de comandos propios más o menos potentes, según el monitor del que se trate.

Para finalizar este capítulo no estará de más el siguiente esquema general de la distribución de la memoria de su C-64.

(1) Página cero. Ahí se encuentran los apuntadores BASIC así como algunas variables del sistema. Hay algunas posiciones no utilizadas por el sistema y que le pueden servir para guardar datos protegidos por el comando NEW. Son las siguientes: 2, 678-767, 820-827, 828-1023 (siempre que no utilice el cassette) o 2024-2039.

(2) Resto de variables, con algunos registros libres que ya se han incluido en (1). 1024-2023 RAM de vídeo (pantalla).

(3) Area BASIC.

(4) a.RAM que está solapada con la ROM del intérprete BASIC. Si desconectamos el intérprete podremos trabajar en esta zona, pero el ordenador ya no podrá entender las palabras BASIC.

(4) b. Interpretador BASIC.

(5) Area libre. Es el lugar ideal, aunque no el único, para guardar los programas en lenguaje máquina.

(6) a. Area libre pero solapada con la ROM de caracteres y la ROM de entrada/salida.

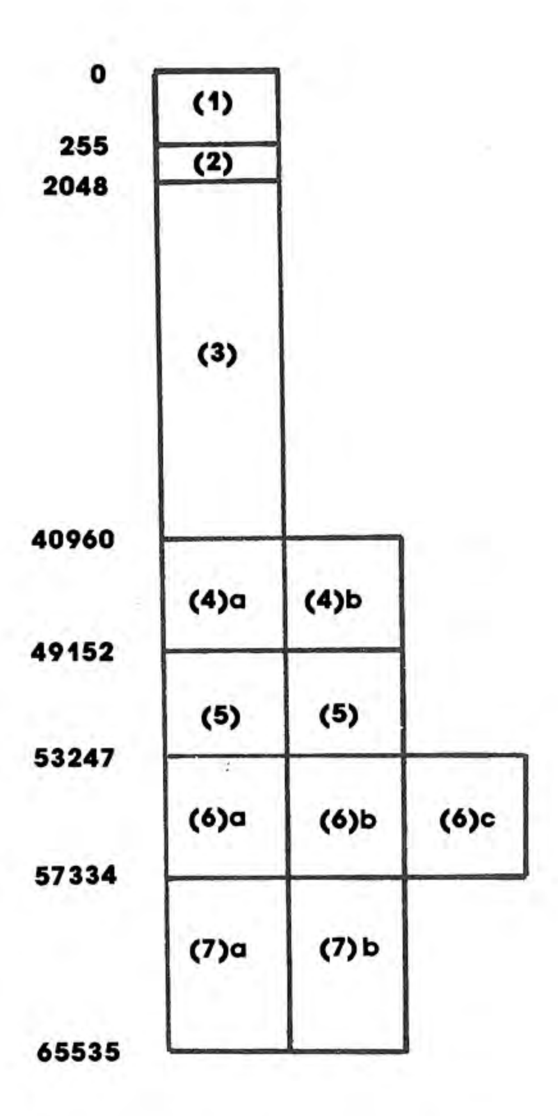
Para trabajar en esta zona habrá que desconectar las dos últimas.

(6) b. ROM de caracteres.

(6) c. Area de entrada/salida.

(7) a. Area libre debajo de la ROM del sistema operativo.

(7) b. Sistema operativo.


0
(1)
255
(2)
2048
(3)
40960
(4)a
(4)b
49152
(5)
(5)
53247
(6)a
(6)b
(6)c
57334
(7)a
(7)b
65535

PROGRAMAS

EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS

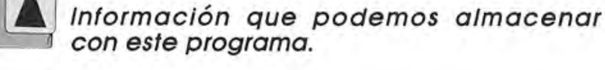


Programa: Agenda telefónica



ESTE programa, que ya apareció en su versión para IBM en un tomo anterior, nos va a permitir la creación de nuestra propia agenda de teléfonos con los siguientes datos:





Información que podemos almacenar con este programa.

— Nombre de la persona.
— Apellidos de la persona.
— Dirección.
— Ciudad.
— Código postal.
— País.
— Número de teléfono.

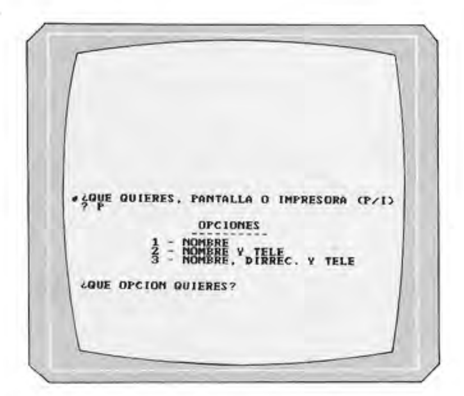
Con este programa podremos introducir nuevas personas con su dirección y su teléfono, buscar datos sobre una de ellas y cambiarlos, buscar todas las personas cuyo nombre empiece por una determinada letra, borrar a una persona y listar todas las personas por la impresora.



Menú del programa con las distintas opciones que ofrece.

Si queremos imprimir todas las personas por impresora, podemos hacerlo de tres maneras:

— Sólo el nombre.
— El nombre y el teléfono.
— El nombre, la dirección y el teléfono.





Menú de impresión.

El programa que se vio en la versión para IBM utilizaba ficheros de acceso directo. Este utiliza ficheros secuenciales para que el programa pueda servir para todos los demás ordenadores. No hay que decir que el programa también funciona en el IBM sin necesidad de realizar ningún cambio.

```
1000 REM ********************************
1010 REM *                              *
1020 REM *      AGENDA TELEFONICA       *
1030 REM *                              *
1040 REM *  POR: PETER BERGMANN         *
1050 REM *                              *
1060 REM ********************************
1070 REM *                              *
1080 REM *          VARIABLES           *
1090 REM *          ---------           *
1100 REM *                              *
1110 REM * T$     ARRAY DE PERSONAS     *
1120 REM * F$     NOMBRE DEL FICHERO    *
1130 REM * N$     REPUESTA FICHERO NUEVO  *
1140 REM * M$     RESPUESTA (S/N)       *
1150 REM * I      NUMERO DE REGISTROS   *
1160 REM * M      OPCION DEL MENU       *
1170 REM * J,K,L CONTADORES             *
1180 REM * W$,E$ APELLIDO Y NOMBRE      *
1190 REM * C$     RESPUESTA PARA CONTINUAR*
1200 REM * S$,J$ MANIPULADORES DE STRING *
1210 REM * L,Q    CONTADORES            *
1220 REM * X$     FLAG PARA REGISTROS   *
1230 REM * J      NUMERO DE REGISTRO    *
1240 REM * JT     PUNTERO REGISTRO ARRIBA *
1250 REM * JM     PUNTERO REGISTRO MEDIO  *
1260 REM * JB     PUNTERO REGISTRO ABAJO  *
1270 REM * H$     LETRA PARA BUSCAR     *
1280 REM * P      NUMERO DE PAGINAS     *
1290 REM * G$     OPCION PANTALLA/IMPRES. *
1300 REM * U      OPCION DE IMPRESION   *
1310 REM * N      NUMERO DE REGISTROS   *
1320 REM *                              *
1330 REM ********************************
1340 REM
1350 REM *** CABECERA ***
1360 REM
1370 KEY OFF
1380 CLS
1390 PRINT "****************************************"
1400 FOR I = 1 TO 19
1410    PRINT  "*"; TAB(40); "*"
1420 NEXT I
1430 PRINT "****************************************"
1440 LOCATE 8,12: PRINT "AGENDA TELEFONICA
1450 LOCATE 9,11: PRINT "-----------------"
1460 LOCATE 12,7: PRINT "(c) Ed. Siglo Cultural, 1987"
1470 REM
1480 REM *** DECLARACION DE ARRAYS E INICIALIZACION ***
1490 REM
1500 DIM T$(200,6)
1510 FOR I = 1 TO 200
1520   FOR J = 1 TO 6
1530       LET T$(I,J) = " "
1540   NEXT J
1550 NEXT I
1560 REM
1570 REM ******************************
1580 REM * NOMBRE DEL FICHERO A USAR ***
1590 REM ******************************
1600 REM
```

```
1610 CLS
1620 LOCATE 2,1: PRINT "(COMO SE LLAMA EL FICHERO ";
1630 INPUT F$
1640 LOCATE 4,1: PRINT "EL NOMBRE ES ";F$
1650 LOCATE 6,1: PRINT "(ES UN FICHERO NUEVO (S/N)";
1660 INPUT N$
1670 IF N$ = "N" THEN GOTO 1690
1680 IF N$ <> "S" THEN GOTO 1650
1690 LOCATE 8,1: PRINT "(ESTA CORRECTO (S/N)";
1700 INPUT M$
1710 IF M$ = "N" THEN GOTO 1610
1720 IF M$ <> "S" THEN GOTO 1690
1730 LET I = 0
1740 IF N$ = "S" THEN GOTO 1990
1750 REM
1760 REM *********************************
1770 REM * LECTURA DEL FICHERO EXISTENTE *
1780 REM *********************************
1790 REM
1800 CLS
1810 PRINT "PON EL CASETE EN PLAY Y ..."
1820 PRINT: PRINT
1830 PRINT "PULSA ENTER";
1840 LET C$=INPUT$(1)
1850 CLS
1860 PRINT "LEYENDO EL FICHERO ";F$
1870 OPEN F$ FOR INPUT AS #1
1880 IF EOF(1) THEN GOTO 1930
1890 LET I = I + 1
1900 FOR J = 1 TO 6
1910    LINE INPUT #1, T$(I,J)
1920 NEXT J
1930 CLOSE #1
1940 CLS
1950 PRINT "FICHERO LEIDO"
1960 PRINT: PRINT
1970 PRINT "PULSA ENTER"
1980 LET C$=INPUT$(1)
1990 REM
2000 REM *******************
2010 REM ***   M E N U   ***
2020 REM *******************
2030 REM
2040 CLS
2050 LOCATE 3,18: PRINT "MENU"
2060 LOCATE 4,17: PRINT "------"
2070 LOCATE 6,12: PRINT "1 - INTRODUCIR PERSONA"
2080 LOCATE 7,12: PRINT "2 - BUSCAR/CAMBIAR PERSONA"
2090 LOCATE 8,12: PRINT "3 - BUSQUEDA"
2100 LOCATE 9,12: PRINT "4 - BORRAR PERSONA"
2110 LOCATE 10,12: PRINT "5 - LISTAR TODAS"
2120 LOCATE 11,12: PRINT "6 - SALIR"
2130 LOCATE 14,1: PRINT "(QUE ELIGES (1-6) ";
2140 LET M=VAL(INPUT$(1))
2150 IF (M < 1) OR (M > 6) GOTO 2030
2160 ON M GOSUB 2460,3380,3950,4220,3800,2190
2170 GOTO 2040
2180 REM
2190 REM ************************
2200 REM * TERMINAR EL PROGRAMA *
2210 REM ************************
2220 REM
2230 CLS
2240 PRINT "PREPARA EL CASETE PARA GRABAR Y ..."
2250 PRINT: PRINT
2260 PRINT "PULSA ENTER"
2270 LET C$=INPUT$(1)
2280 CLS
2290 PRINT "GRABANDO EN EL FICHERO ";F$
```

```
2300 OPEN F$ FOR OUTPUT AS #1
2310 FOR J = 1 TO I
2320    IF T$(J,1) = " " THEN GOTO 2360
2330    FOR K = 1 TO 6
2340       PRINT #1, T$(J,K)
2350    NEXT K
2360 NEXT J
2370 CLOSE #1
2380 CLS
2390 PRINT "INFORMACION GRABADA EN EL FICHERO ";F$
2400 PRINT
2410 PRINT
2420 PRINT "ADIOS..."
2430 KEY ON
2440 END
2450 REM
2460 REM **********************
2470 REM * INTRODUCIR PERSONA *
2480 REM **********************
2490 REM
2500 CLS
2510 PRINT "PARA SALIR PULSE 'X' EN EL NOMBRE"
2520 PRINT
2530 LINE INPUT "NOMBRE    : ";W$
2540 IF W$ = "X" THEN GOTO 2670
2550 IF W$ = "" THEN GOTO 2500
2560 LINE INPUT "APELLIDOS: ";E$
2570 GOSUB 2730
2580 GOSUB 3240
2590 IF X$ = "D" GOTO 2680
2600 LINE INPUT "DIRECCION: ";T$(J,2)
2610 LINE INPUT "CIUDAD    : ";T$(J,3)
2620 LINE INPUT "CODIGO    : ";T$(J,4)
2630 LINE INPUT "PAIS      : ";T$(J,5)
2640 LINE INPUT "TELEFONO : ";T$(J,6)
2650 IF I = 201 THEN PRINT "FICHERO LLENO": GOTO 2670
2660 GOTO 2500
2670 RETURN
2680 CLS
2690 PRINT "FICHA IGNORADA";:INPUT C$
2700 GOTO 2460
2710 END
2720 REM
2730 REM ****************
2740 REM * ASSEMBLE KEY *
2750 REM ****************
2760 REM
2770 LET S$ = " "
2780 LET W$ = W$ + S$
2790 LET W$ = W$ + E$
2800 RETURN
2810 REM
2820 REM **************
2830 REM * DISECT KEY *
2840 REM **************
2850 REM
2860 LET S$ = T$(J,1)
2870 LET L = LEN(S$)
2880 LET Q = INSTR(S$ , " ")
2890 LET L = L - Q
2900 LET Q = Q - 1
2910 LET W$ = LEFT$(S$,Q)
2920 LET E$ = RIGHT$(S$,L)
2930 RETURN
2940 REM
2950 REM **********************
2960 REM * FICHA YA EXISTENTE *
2970 REM **********************
```

```
2980 REM
2990 LET X$ = "D"
3000 REM
3010 RETURN
3020 REM
3030 REM *******************
3040 REM * BUSCAR REGISTROS *
3050 REM *******************
3060 REM
3070 LET X$ = "N"
3080 IF (W$ < T$(1,1)) OR (I = 0) THEN LET J = 1: GOTO 3220
3090 IF W$ > T$(I,1) THEN LET J = I + 1: GOTO 3220
3100 IF W$ = T$(1,1) THEN LET J = 1: GOSUB 2950: GOTO 3220
3110 IF W$ = T$(I,1) THEN LET J = I: GOSUB 2950: GOTO 3220
3120 LET JT = I
3130 IF I > 0 THEN LET JM = INT((I+1)/2)
3140 LET JB = 1
3150 IF X$ <> "N" THEN GOTO 3210
3160   IF W$ > T$(JM,1) THEN LET JB = JM
3170   IF W$ < T$(JM,1) THEN LET JT = JM
3180   IF W$ = T$(JM,1) THEN GOSUB 2950: GOTO 3210
3190   LET JM = INT((JT + JB + 1)/2)
3200   IF (JM = JT) OR (JM = JB) THEN LET X$ = "F"
3210 LET J = JM
3220 RETURN
3230 REM
3240 REM ***************************
3250 REM * INSERTAR NUEVO REGISTRO *
3260 REM ***************************
3270 REM
3280 GOSUB 3030
3290 IF X$ = "D" THEN GOTO 3370
3300 LET I = I + 1
3310 FOR K = I TO J STEP -1
3320    FOR L = 1 TO 6
3330       LET T$(K,L) = T$(K-1,L)
3340    NEXT L
3350 NEXT K
3360 LET T$(J,1) = W$
3370 RETURN
3380 REM
3390 REM *********************
3400 REM * LOCALIZAR PERSONA *
3410 REM *********************
3420 REM
3430 CLS
3440 LOCATE 3,14: PRINT "LOCALIZAR PERSONA"
3450 LOCATE 5,1: PRINT "(NOMBRE   : ";
3460 INPUT W$
3470 LOCATE 6,1: PRINT "(APELLIDOS: ";
3480 INPUT E$
3490 GOSUB 2730
3500 GOSUB 3030
3510 CLS
3520 IF X$ <> "D" THEN LOCATE 7,12: PRINT "FICHA NO ENCONTRADA": GOTO 3720
3530 GOSUB 2820
3540 LOCATE 2,2: PRINT "NOMBRE ";W$
3550 LOCATE 3,2: PRINT "       ";E$
3560 LOCATE 4,2: PRINT "DIREC. ";T$(J,2)
3570 LOCATE 5,2: PRINT "CIUDAD ";T$(J,3)
3580 LOCATE 6,2: PRINT "CODIGO ";T$(J,4)
3590 LOCATE 7,2: PRINT "PAIS   ";T$(J,5)
3600 LOCATE 8,2: PRINT "TELEF. ";T$(J,6)
3610 IF M <> 2 THEN GOTO 3770
3620 LOCATE 14,2: PRINT "(QUIERES CAMBIAR ESTA FICHA (S/N)";
3630 LET M$=INPUT$(1)
3640 IF M$ = "S" THEN GOSUB 4490: GOTO 3530
3650 IF M$ <> "N" GOTO 3620
```

```
3660 LOCATE 14,2: PRINT SPACE$(38)
3670 LOCATE 14,2: PRINT "(QUIERES IMPRIMIR ESTA FICHA (S/N)";
3680 LET M$=INPUT$(1)
3690 IF M$ = "S" THEN GOSUB 4640: GOTO 3710
3700 IF M$ <> "N" GOTO 3660
3710 LOCATE 14,2: PRINT SPACE$(38)
3720 IF M <> 2 THEN GOTO 3770
3730 LOCATE 14,2: PRINT "(BUSCAMOS MAS (S/N)";
3740 LET M$=INPUT$(1)
3750 IF M$ = "S" GOTO 3430
3760 IF M$ <> "N" GOTO 3730
3770 RETURN
3780 REM
3790 REM ************
3800 REM * IMPRIMIR *
3810 REM ************
3820 REM
3830 GOSUB 4760
3840 LET P = 1
3850 CLS
3860 LOCATE 1,14: PRINT "LISTADO DE NOMBRES"
3870 FOR J = 1 TO I
3880    IF M <> 3 THEN GOSUB 5000
3890    IF M = 3 THEN GOSUB 4150
3900 NEXT J
3910 PRINT
3920 PRINT "PULSA ENTER"
3930 LET C$=INPUT$(1)
3940 RETURN
3950 REM
3960 REM **************************
3970 REM * BUSQUEDA POR UNA LETRA *
3980 REM **************************
3990 REM
4000 CLS
4010 LOCATE 2,12: PRINT "BUSQUEDA POR LETRA"
4020 LOCATE 4,1: PRINT "PRIMERA LETRA DEL NOMBRE ";
4030 INPUT H$
4040 CLS
4050 LOCATE 2,16: PRINT "NOMBRES CON ";H$
4060 PRINT :PRINT
4070 GOSUB 3870
4080 PRINT : PRINT
4090 PRINT "(BUSCAMOS MAS (S/N):";
4100 LET M$=INPUT$(1)
4110 IF M$ = "S" THEN GOTO 3990
4120 IF M$ <> "N" THEN GOTO 4090
4130 RETURN
4140 REM
4150 REM **************************
4160 REM * CHEQUEAR PRIMERA LETRA *
4170 REM **************************
4180 REM
4190 LET J$ = LEFT$(T$(J,1),1)
4200 IF J$ = H$ THEN GOSUB 2820: PRINT W$;", ";E$
4210 RETURN
4220 REM
4230 REM ****************
4240 REM * BORRAR FICHA *
4250 REM ****************
4260 REM
4270 CLS
4280 LOCATE 3,11: PRINT "BORRAR PERSONA"
4290 GOSUB 3450
4300 IF X$ = "D" THEN GOTO 4360
4310 LOCATE 12,1: PRINT "(BORRAMOS MAS (S/N)";
4320 LET M$=INPUT$(1)
4330 IF M$ = "S" GOTO 4270
```

```
4340 IF M$ <> "N" GOTO 4310
4350 RETURN
4360 LOCATE 14,2: PRINT "(ESTAS SEGURO (S/N)";
4370 INPUT M$
4380 IF M$ = "N" THEN CLS: GOTO 4310
4390 IF M$ <> "S" THEN GOTO 4360
4400 FOR K = J TO I
4410    FOR L = 1 TO 6
4420       LET T$(K,L) = T$(K+1,L)
4430    NEXT L
4440 NEXT K
4450 LET I = I - 1
4460 CLS
4470 LOCATE 4,12: PRINT "FICHA BORRADA!"
4480 GOTO 4310
4490 REM
4500 REM ***********************
4510 REM * CAMBIAR UN REGISTRO *
4520 REM ***********************
4530 REM
4540 LOCATE 14,2: PRINT SPACE$(38)
4550 LOCATE 14,2: PRINT "(QUE QUIERES CAMBIAR (D,C,Z,P,T)";
4560 LET M$=INPUT$(1)
4570 IF M$ = "D" THEN LOCATE 4,9: LINE INPUT T$(J,2)
4580 IF M$ = "C" THEN LOCATE 5,9: LINE INPUT T$(J,3)
4590 IF M$ = "Z" THEN LOCATE 6,9: LINE INPUT T$(J,4)
4600 IF M$ = "P" THEN LOCATE 7,9: LINE INPUT T$(J,5)
4610 IF M$ = "T" THEN LOCATE 8,9: LINE INPUT T$(J,6)
4620 CLS
4630 RETURN
4640 REM
4650 REM ************
4660 REM * IMPRIMIR *
4670 REM ************
4680 REM
4690 GOSUB 5390
4700 LPRINT W$;", ";E$
4710 LPRINT T$(J,2)
4720 LPRINT T$(J,3);"  ";T$(J,4)
4730 LPRINT T$(J,5)
4740 LPRINT T$(J,6)
4750 RETURN
4760 REM
4770 REM ************************
4780 REM * PANTALLA O IMPRESORA *
4790 REM ************************
4800 REM
4810 CLS
4820 LOCATE 2,1: PRINT "(QUE QUIERES, PANTALLA O IMPRESORA (P/I)";
4830 LET G$=INPUT$(1)
4840 IF (G$ <> "P") AND (G$ <> "I") THEN GOTO 4810
4850 LOCATE 5,16: PRINT "OPCIONES"
4860 LOCATE 6,15: PRINT "----------"
4870 LOCATE 7,10: PRINT "1 - NOMBRE"
4880 LOCATE 8,10: PRINT "2 - NOMBRE Y TELE"
4890 LOCATE 9,10: PRINT "3 - NOMBRE, DIRREC. Y TELE"
4900 LOCATE 12,1: PRINT "(QUE OPCION QUIERES";
4910 INPUT U
4920 IF (U < 1) OR (U > 3) THEN GOTO 4900
4930 LOCATE 14,1: PRINT "(ESTAS SEGURO (S/N)";
4940 LET M$=INPUT$(1)
4950 IF M$ = "N" THEN GOTO 4810
4960 IF M$ <> "S" THEN GOTO 4930
4970 LET N = 0
4980 IF G$ = "I" THEN GOSUB 5390
4990 RETURN
5000 REM
5010 REM ************************
```

```
5020 REM * IMPRESION EN PANTALLA *
5030 REM *************************
5040 REM
5050 GOSUB 2820
5060 IF G$ = "I" THEN GOTO 5210
5070 PRINT W$;", "; E$
5080 IF U = 3 THEN PRINT T$(J,2): PRINT T$(J,3);T$(J,4): PRINT T$(J,5)
5090 IF U > 1 THEN PRINT T$(J,6)
5100 PRINT
5110 LET N = N + 1
5120 IF (U = 1) AND (N/P = 10) THEN GOTO 5160
5130 IF (U = 2) AND (N/P = 7) THEN GOTO 5160
5140 IF (U = 3) AND (N/P = 3) THEN GOTO 5160
5150 GOTO 5380
5160 LET P = P + 1
5170 PRINT: PRINT "PULSA ENTER";
5180 LET C$ = INPUT$(1)
5190 CLS
5200 GOTO 5380
5210 REM
5220 REM *************************
5230 REM * IMPRIMIR EN IMPRESORA *
5240 REM *************************
5250 REM
5260 LPRINT W$; E$
5270 IF U = 3 THEN LPRINT T$(J,2): LPRINT T$(J,3),T$(J,4): LPRINT T$(J,5)
5280 IF U > 1 THEN LPRINT T$(J,6)
5290 LPRINT
5300 LET N = N + 1
5310 IF (U = 1) AND (N/P = 30) THEN GOTO 5350
5320 IF (U = 2) AND (N/P = 20) THEN GOTO 5350
5330 IF (U = 3) AND (N/P = 10) THEN GOTO 5350
5340 GOTO 5380
5350 LET P = P + 1
5360 LPRINT CHR$(12)
5370 CLS
5380 RETURN
5390 REM
5400 REM *********************
5410 REM * PREPARA IMPRESORA *
5420 REM *********************
5430 REM
5440 LPRINT CHR$(27)+CHR$(64)
5450 CLS
5460 LOCATE 2,1: PRINT "(QUIERES IMPRESION GRANDE O PEQUENA (G/P)";
5470 LET M$=INPUT$(1)
5480 IF M$ = "P" THEN LPRINT CHR$(27)+CHR$(15)
5490 IF M$ = "G" THEN LPRINT CHR$(18)
5500 IF (M$ <> "P") AND (M$ <> "G") THEN GOTO 5450
5510 CLS
5520 LOCATE 2,1: PRINT "PREPARA EL PAPEL Y PULSA RETURN";
5530 LET C$ = INPUT$(1)
5540 RETURN
```

Para que el programa funcione en el COMMODORE, AMSTRAD y MSX hay que realizar los siguientes cambios:

COMMODORE:

```
1370 REM
1380 PRINT CHR$(147)
1440 POKE 214,7:POKE 211,11:PRINT "AGENDA TELEFONICA"
1450 POKE 214,8:POKE 211,10: PRINT "-----------------"
1460 POKE 214,11:POKE 211,6:PRINT "(c) Ed. Siglo Cultural, 1987"
1610 PRINT CHR$(147)
```

```
1620 POKE 214,1:POKE 211,0:PRINT "COMO SE LLA-
     MA EL FICHERO'
1640 POKE 214,3:POKE 211,0:PRINT "EL NOMBRE
     ES";F$
1650 POKE 214,5:POKE 211,0:PRINT "ES UN FICHERO
     NUEVO (S/N)";
1690 POKE 214,7:POKE 211,0:PRINT "ESTA CORREC-
     TO (S/N)";
1800 PRINT CHR$(147)
1840 GET C$:IF C$="" THEN GOTO 1840
1850 PRINT CHR$(147)
1870 OPEN 1,1,0,F$
1880 INPUT#1,I
1890 FOR K=1 TO I
1925 NEXT K
1930 CLOSE 1
1980 GET C$:IF C$="" THEN GOTO 1980
2040 PRINT CHR$(147)
2050 POKE 214,2:POKE 211,17:PRINT "MENU"
2060 POKE 214,3:POKE 211,16:PRINT "------"
2070 POKE 214,5:POKE 211,11:PRINT "1 - INTRODU-
     CIR PERSONA"
2080 POKE 214,6:POKE 211,11:PRINT "2 - BUS-
     CAR/CAMBIAR PERSONA"
2090 POKE 214,7:POKE 211,11:PRINT "3 - BUSQUE-
     DA"
2100 POKE 214,8:POKE 211,11:PRINT "4 - BORRAR
     PERSONA"
2110 POKE 214,9:POKE 211,11:PRINT "5 - LISTAR TO-
     DAS"
2120 POKE 214,10:POKE 211,11:PRINT "6 - SALIR"
2130 POKE 214,13:POKE 211,0:PRINT "QUE ELIGES
     (1-6) ";
2140 GET C$:IF C$="" THEN GOTO 2140
2145 LET M=VAL(C$)
2230 PRINT CHR$(147)
2270 GET C$=:IF C$"" THEN GOTO 2270
2280 PRINT CHR$(147)
2300 OPEN 1,1,1,F$
2305 PRINT #1,I
2370 CLOSE 1
2380 PRNT CHR$(147)
2430 REM
2500 PRINT CHR$(147)
2680 PRINT CHR$(147)
2880 FOR W1=1 TO LEN(S$)
2882 IF MID$(S$,W1,1)=" " THEN LET Q=W1:LET
     W1=LEN(S$)
2884 NEXT W1
3430 PRINT CHR$(147)
3440 POKE 214,2:POKE 211,13:PRINT "LOCALIZAR
     PERSONA"
3450 POKE 214,4:POKE 211,0:PRINT "NOMBRE : ";
3470 POKE 214,5:POKE 211,0:PRINT "APELLIDOS : ";
3510 PRINT CHR$(147)
3520 IF X$<>"D" THEN POKE 214,6:POKE
     211,11:PRINT "FICHA NO ENCONTRADA":GO-
     TO 3720
3540 POKE 214,1:POKE 211,1:PRINT "NOMBRE ";W$
3550 POKE 214,2:POKE 211,1:PRINT "         ";E$
3560 POKE 214,3:POKE 211,1:PRINT "DIREC.
     ";T$(J,2)
3570 POKE 214,4:POKE 211,1:PRINT "CIUDAD
     ";T$(J,3)
3580 POKE 214,5:POKE 211,1:PRINT "CODIGO
     ";T$(J,4)
3590 POKE 214,6:POKE 211,1:PRINT "PAIS   ";T$(J,5)
3600 POKE 214,7:POKE 211,1:PRINT "TELEF. ";T$(J,6)
3620 POKE 214,13:POKE 211,2:PRINT "QUIERES CAM-
     BIAR ESTA FICHA (S/N)";
3630 GET M$:IF M$="" THEN GOTO 3630
3660 POKE 214,13:POKE 211,1:FOR W1=1 TO
     38:PRINT " ";:NEXT W1
3670 POKE 214,13:POKE 211,1:PRINT "QUIERES IM-
     PRIMIR ESTA FICHA (S/N)";
3680 GET M$:IF M$="" THEN GOTO 3680
3710 POKE 214,13:POKE 211,1:FOR W1=1 TO
     38:PRINT " ";: NEXT W1
3730 POKE214,13:POKE 211,1:PRINT "BUSCAMOS
     MAS (S/N)";
3740 GET M$:IF M$="" THEN GOTO 3740
3850 PRINT CHR$(147)
3860 POKE 214,0:POKE 211,13:PRINT "LISTADO DE
     NOMBRES"
3865 OPEN 1,4:CMD 1
3905 CLOSE 1
3930 GET C$:IF C$="" THEN GOTO 3930
4000 PRINT CHR$(147)
4010 POKE 214,1:POKE 211,11:PRINT "BUSQUEDA
     POR LETRA"
4020 POKE 214,3:POKE 211,0:PRINT "PRIMERA LETRA
     DEL NOMBRE ";
4040 PRINT CHR$(147)
4050 POKE 214,1:POKE 211,15:PRINT "NOMBRES
     CON ";H$
4100 GET M$:IF M$="" THEN GOTO 4100
4270 PRINT CHR$(147)
4280 POKE 214,2:POKE 211,10:PRINT "BORRAMOS
     PERSONA"
4310 POKE 214,11:POKE 211,0:PRINT "BORRAMOS
     MAS (S/N)"
4320 GET M$:IF M$="" THEN GOTO 4320
4360 POKE 214,13:POKE 211,1:PRINT "ESTAS SEGURO
     (S/N)"
4380 IF M$="N" THEN PRINT CHR$(147):GOTO 4310
4460 PRINT CHR$(147)
4470 POKE 214,3:POKE 211,11:PRINT "FICHA BORRA-
     DA"
4540 POKE 214,13:POKE 211,1:FOR W1=1 TO
     38:PRINT " ";:NEXT W1
4550 POKE 214,13:POKE 211,1:PRINT "QUE QUIERES
     CAMBIAR (D,C,Z,P,T)";
4560 GET M$:IF M$="" THEN GOTO 4560
4620 PRINT CHR$(147)
4700 PRINT W$;", ";E$
4710 PRINT T$(J,2)
4720 PRINT T$(J,3);"      ";T$(J,4)
4730 PRINT T$(J,5)
4740 PRINT T$(J,6)
4810 PRINT CHR$(147)
4820 POKE 214,1:POKE 211,0:PRINT "QUE QUIERES
     PANTALLA O IMPRESORA (P/I)";
4830 GET G$:IF G$="" THEN GOTO 4830
4850 POKE 214,4:POKE 211,15:PRINT "OPCIONES"
4860 POKE 214,5:POKE 211,14:PRINT "----------"
4870 POKE 214,6:POKE 211,9:PRINT "1 - NOMBRE"
4880 POKE 214,7:POKE 211,9:PRINT "2 - NOMBRE Y
     TELF."
4890 POKE 214,8:POKE 211,9:PRINT "3 - NOMBRE,
     DIR. Y TELEF."
4900 POKE 214,11:POKE 211,0:PRINT "QUE OPCION
     QUIERES ";
4930 POKE 214,13:POKE 211,0:PRINT "ESTAS SEGURO
     (S/N)"
4940 GET M$:IF M$="" THEN GOTO 4940
5180 GET C$:IF C$="" THEN GOTO 5180
5190 PRINT CHR$147)
5260 PRINT W$;E$
```

```
5270 IF U=3 THEN PRINT T$(J,2):PRINT T$(J,3),
     T$(J,4):PRINT T$(J,5)
5280 IF U>1 THEN PRINT T$(J,6)
5290 PRINT
5360 PRINT CHR$(12)
5370 PRINT CHR$(147)
5440 REM
5450 REM
5460 REM
5470 REM
5480 REM
5490 REM
5500 REM
5510 PRINT CHR$(147)
5520 POKE 214,1:POKE 211,0:PRINT "PREPARA EL PA-
     PEL Y PULSA RETURN";
5530 GET C$:IF C$="" THEN GOTO 5530
```

MSX:

Sólo hay que cambiar de orden los argumentos de todas las sentencias LOCATE. Si, por ejemplo, vemos la línea 2050 que pone:

```
2050 LOCATE 3,18:PRINT "MENU"
```

nosotros tendremos que poner:

```
2050 LOCATE 18,3:PRINT "MENU"
```

AMSTRAD:

Aparte de tener que cambiar el orden de todas las sentencias LOCATE tal y como se explica en las variaciones para MSX, hay que realizar los siguientes cambios:

```
1870 OPENIN F$
1880 IF EOF THEN GOTO 1930
1910 LINE INPUT #9,T$(I,J)
1930 CLOSE
2300 OPENOUT F$
2340 PRINT #9,T$(J,K)
2370 CLOSE
4700 PRINT #8,W$;", ";E$
4710 PRINT #8,T$(J,2)
4720 PRINT #8,T$(J,3);"        ";T$(J,4)
4730 PRINT #8,T$(J,5)
4740 PRINT #8,T$(J,6)
5260 PRINT #8,W$;E$
5270 IF U=3 THEN PRINT #8,T$(J,2):PRINT
     #8,T$(J,3),T$(J,4):PRINT #8,T$(J,5)
5280 IF U>1 THEN PRINT #8,T$(J,6)
5290 PRINT #8
5360 PRINT #8,CHR$(12)
5440 REM
5450 REM
5460 REM
5470 REM
5480 REM
5490 REM
5500 REM
```



Programa: «El ahorcado» para SPECTRUM

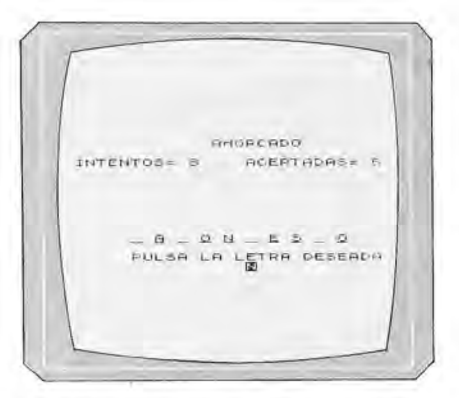
El programa que vamos a ver a continuación es de todos conocido y a duras penas necesita de explicación. Este programa está realizado en versión para SPECTRUM, ya que anteriormente apareció para otros ordenadores.





Pantalla de presentación del programa «Ahorcado».

El juego, en resumen, consiste en adivinar una palabra que ha pensado el ordenador. El usuario irá introduciendo las letras que él cree que pueden componer la palabra una a una. Si la letra es correcta, el ordenador la colocará en su sitio y si no lo es, restará uno al número de oportunidades que tiene el usuario para acertar la palabra.





El programa en plena ejecución.

El juego termina cuando el jugador ha acertado la palabra o cuando ha agotado las oportunidades que se le ofrecen.

```
  10 REM ***********************
  20 REM ****** AHORCADO *******
  30 REM ***********************
  40 REM **** CARLOS DORAL *****
  50 REM ***********************
  60 BORDER 7
  70 PAPER 7
  80 INK 0
  90 CLS
 100 DIM a(20)
 110 PRINT AT 3,12; PAPER 2; INK
 7; BRIGHT 1; FLASH 1;"AHORCADO"
 120 PRINT AT 5,0;"  Adivina la
palabra que piensa el ordenador
antes de que se te acaben las op
ortunidades."
 130 LET a$="    INFORMATICA PAS
O A PASO     "
 140 PRINT  FLASH 1;AT 21,2;"PUL
SA UNA TECLA PARA EMPEZAR"
 150 PRINT AT 10,0;a$
 160 LET a$=a$(2 TO )+a$(1)
 170 FOR f=1 TO 10
 180 NEXT f
 190 IF INKEY$="" THEN  GO TO 15
0
 200 LET c=1
 210 LET bi=0
 220 CLS
 230 RANDOMIZE PEEK 23672
 240 LET rn=1+INT (RND*19)
 250 FOR f=1 TO 20
 260     IF a(f)=rn THEN  GO TO 2
40
 270 NEXT f
 280 LET a(rn)=rn
 290 LET da=730+(rn*10)
 295 RESTORE da
 300 READ a$
 310 LET b$=a$
 320 PRINT AT 0,12; BRIGHT 1; FL
ASH 1; INK 1; PAPER 7;"AHORCADO"
 330 LET t=0
 340 LET b=0
 350 PRINT AT 5,2; FLASH 1;" * A
TENCION QUE EMPEZAMOS * "
 360 FOR f=1 TO 6
 370     BEEP .20,20
 380 NEXT f
 390 FOR f=5 TO 3+(LEN a$*2) STE
P 2
 400     PRINT AT 10,f;"_"
 410 NEXT f
 420 PRINT AT 5,2; FLASH 0;"
                           "
 430 FOR w=1 TO LEN a$+4
 440     PRINT AT 2,0;"INTENTOS=
";w-1;"    ACERTADAS= ";b
 450     PRINT AT 12,5;"PULSA LA
LETRA DESEADA"
 460     POKE 23658,8
 470     LET k$=INKEY$
 480     IF CODE k$<65 OR CODE k$
>90 THEN  GO TO 460
 490     PRINT AT 13,15; INVERSE
1;k$
 500     FOR f=1 TO LEN a$
 510         IF k$=a$(f TO f) AND
a$(f TO f)<>" " THEN  PRINT AT 1
0,3+(2*f); OVER 1;k$: LET b=b+1:
 LET a$(f TO f)=" "
 520     NEXT f
 530    IF c=20 THEN  GO TO 1030
 540     IF b=LEN a$ THEN  GO TO
940
 550     FOR f=1 TO 20
 560     NEXT f
 570 NEXT w
 580 PRINT  FLASH 1;AT 15,2;"LO
SIENTO,NO LA HAS ADIVINADO"
 590 FOR f=50 TO 0 STEP -5
 600     BEEP .20,f
 610 NEXT f
 620 PRINT AT 17,3;"LA PALABRA E
RA "; FLASH 1;b$
 630 FOR f=1 TO 300
 640 NEXT f
 650 PRINT AT 20,2;"  VAMOS A PO
R OTRA (S/N) ? "
 660 POKE 23658,8
 670 LET k$=INKEY$
 680 IF k$="S" OR k$="s" THEN  L
ET c=c+1: GO TO 220
 690 IF k$="N" OR k$="n" THEN  G
O TO 1090
 700 GO TO 660
 710 REM ***********************
 720 REM *DATAS DE LAS PALABRAS*
 730 REM ***********************
 740 DATA "COMPUTER"
 750 DATA "VOLUMEN"
 760 DATA "BALONCESTO"
 770 DATA "DISCOTECA"
 780 DATA "INDESCOMP"
 790 DATA "VENTANUCO"
 800 DATA "BISMUTO"
 810 DATA "INSERTAR"
 820 DATA "ORBICULAR"
 830 DATA "CUADRICEPS"
 840 DATA "ENSAMBLADOR"
 850 DATA "CALENDARIO"
 860 DATA "BOLIGRAFO"
 870 DATA "IMPOSIBLE"
 880 DATA "SABOTAJE"
 890 DATA "IZQUIERDA"
 900 DATA "DESPLAZAMIENTO"
 910 DATA "COMENTARIO"
 920 DATA "RANDOMIZE"
 930 DATA "TRIANGULO"
 940 REM ***********************
 950 REM ****** ACERTADA *******
 960 REM ***********************
 970 PRINT AT 20,2; FLASH 1;" MU
Y BIEN,LA HAS ACERTADO !"
 980 FOR f=1 TO 20
 990     BEEP .10,-50+INT (RND*50
)
1000 NEXT f
1010 LET bi=bi+1
1020 GO TO 650
1030 REM ***********************
1040 REM ******** FINAL ********
1050 REM ***********************
1060 PRINT AT 20,1;"SE ME HAN AC
ABADO LAS PALABRAS"
1070 FOR f=1 TO 300
```

```
1080 NEXT f
1090 CLS
1100 PRINT AT 0,12; BRIGHT 1; FL
ASH 1; INK 1; PAPER 7;"AHORCADO"
1110 PRINT AT 4,3;"NUMERO DE PAL
ABRAS= ";c
1120 PRINT AT 6,3;"PALABRAS ACER
TADAS= ";bi
1130 PRINT AT 8,3;"PORCENTAJE AD
IVINATORIO= ";INT ((bi*100)/c);"
%"
1140 PRINT AT 20,2;"PULSA UNA TE
CLA PARA EMPEZAR"
1150 IF INKEY$<>"" THEN  CLS : G
O TO 110
1160 GO TO 1150
```

Al final del juego, el ordenador hace un balance del mismo y nos dice cuántas palabras hemos acertado.



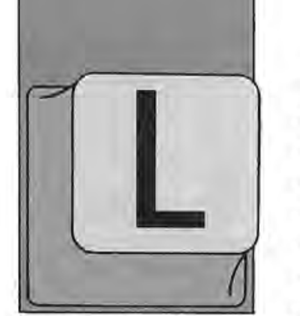


Resultados finales.



TECNICAS DE ANALISIS

DIAGRAMAS



LOS diagramas de proceso (o diagramas de flujo) suponen un esquema sucinto y claro de los pasos sucesivos que se darán en la aplicación, así como de los archivos utilizados y, por tanto, de la evolución de los datos en el proceso. En la figura aparece un diagrama sencillo de proceso. En él se observan los símbolos normalmente utilizados en los diagramas:

Símbolo terminal. Indica el comienzo o final del proceso. Sólo se utiliza en la rama principal del programa, pues el resto de los procesos predefinidos tienen su comienzo en un punto de llamada (que viene indicado por un conector) y su final en un punto de retorno (otro conector).

Conector. Punto donde enlazan las líneas de flujo que provienen de otros procesos. Normalmente suele estar al comienzo de un proceso. Es usual indicar en las líneas que llegan al conector el punto de donde provienen (proceso del que parten), así como la página donde parten), así como la página donde aparece este punto de origen (si el diagrama se extiende a lo largo de varias páginas, como es usual).

Símbolo de decisión. Cuando la decisión a tomar es unitaria (sólo una condición simple). De este punto parten varias líneas de proceso alternativas: la línea principal, que sigue el diagrama y otras que van hacia otros puntos del diagrama.

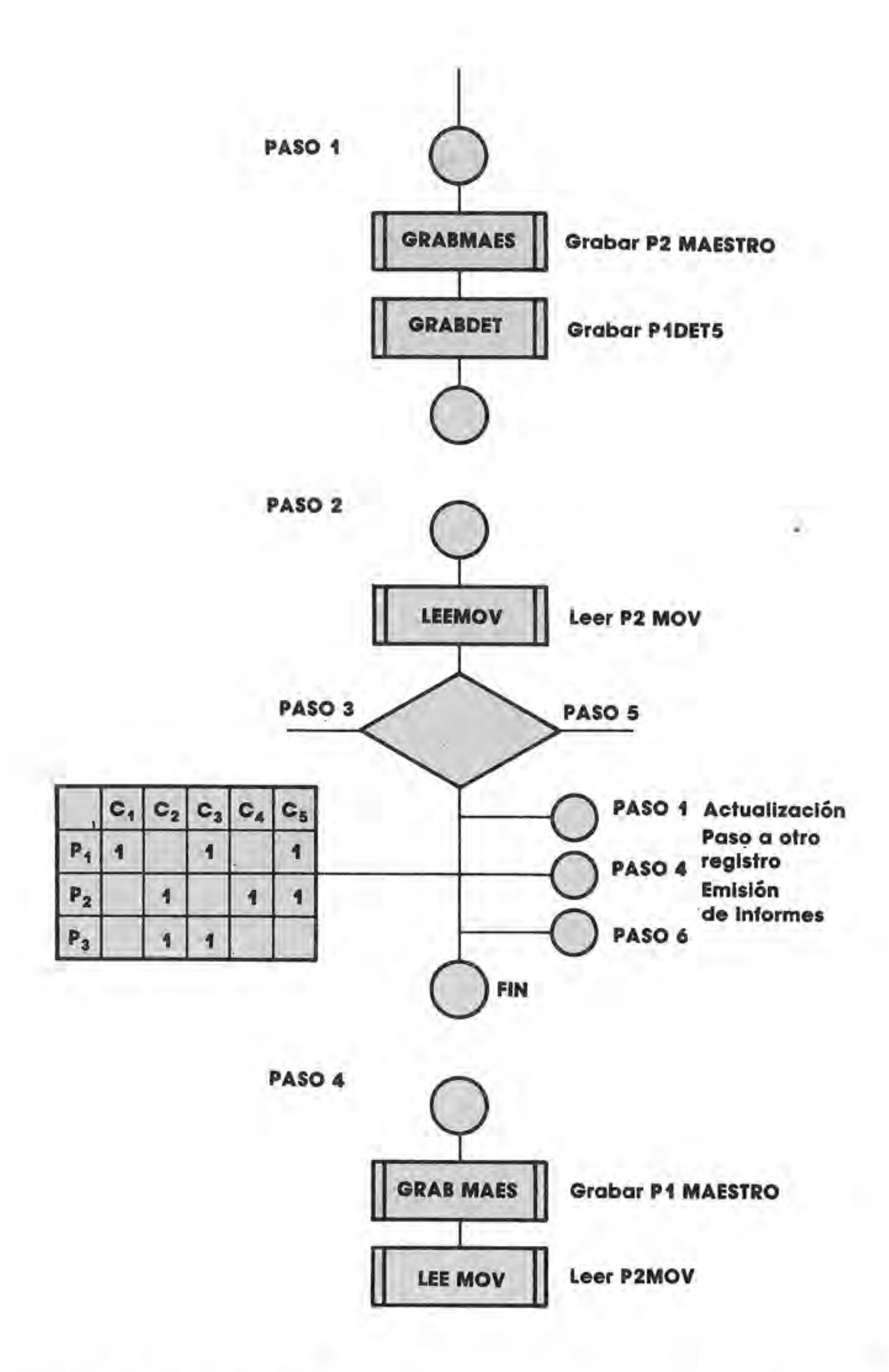
Proceso en general. Cuando el proceso es simple y queda claramente identificado en este punto del diagrama.

Proceso predefinido. Se da el nombre del proceso (o un mnemotécnico propio del diagrama). En otro lugar del diagrama deberá aparecer este proceso bajo el mismo nombre y con un punto de comienzo (o «punto de llamada») y un punto de vuelta a la línea principal de proceso.

En ocasiones hay puntos de la aplicación en los cuales se realizan varios procesos sucesivos, o no se realizan, dependiendo de un conjunto de condiciones a evaluar: en ese caso, aparece una estructura de proceso como la que se muestra en la figura, en el paso 2, que incluye una pequeña tabla (a modo de tabla de decisión) para decidir si los pasos 1, 4, 6, se realizan o no, respectivamente. Usualmente se suelen tener en cuenta en el diseño de estos diagramas las siguientes normas:

a) Los mnemotécnicos utilizados deben ser claros y aludir a las actividades que realmente se realizan en el proceso al que dan nombre.

b) La referencia del punto de entrada de donde viene el control se suele poner a la izquierda del símbolo de conexión; la referencia del punto al que se cede control, debe estar a la derecha del conec-



tor (e incluyendo el número de página donde dicho punto de control sucesivo aparece).

c) Todos los saltos de control dentro del programa deben estar claramente identificados mediante las oportunas referencias; en todos los diagramas relacionados deben aparecer las mismas referencias.

d) El nivel de detalle en el diagrama puede variar, y un elemento de proceso definido puede ser una operación elemental o una subrutina o un proceso más amplio; por razones de coherencia, es importante que, a lo largo de todo el diagrama, se mantenga el mismo nivel de detalle para los diferentes procesos referenciados.

e) En los procesos predefinidos no hay que poner referencia alguna en el punto de entrada ni en el punto de salida, puesto que pueden ser llamados desde distintos procesos en la aplicación. El retorno se hace al punto donde fueron llamados.

f) En un conector debe procurarse

que las líneas que representan los caminos de acceso lleguen por la izquierda y/o por arriba. Las líneas de flujo de salida del conector partirán, preferiblemente, hacia la derecha y hacia abajo. Las referencias se anotarán siguiendo el mismo criterio. Al acabar una página, para seguir en la siguiente (o en la columna de la derecha si el diagrama se dibuja a dos columnas, como es usual) se pondrá un conector con sus referencias y símbolos usuales.

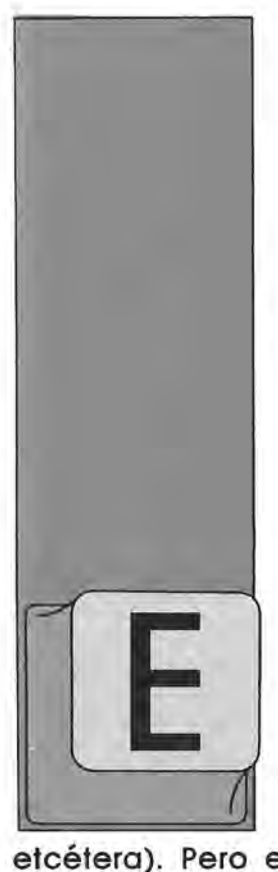
g) Las bifurcaciones por tomas de decisión se señalan preferiblemente mediante el símbolo previsto para este tipo de acción, excepto si existen varias alternativas y se hace necesario poner una pequeña tabla para seleccionar los pasos adecuados.

h) Las anotaciones complementarias y comentarios (incluidos en la caja correspondiente) se reducirán a lo estrictamente imprescindible y se harán, si es necesario incluirlas, lo más concisas posible.

i) Cuando haya pasos de salida alternativos en un proceso, cada línea (representativa de uno de estos procesos alternativos) terminará en un conector hacia la derecha con la referencia de la rutina correspondiente. Esta rutina se reseñará aparte, aunque sea un solo proceso predefinido o una actividad simple.

TECNICAS DE PROGRAMACION

COMPATIBILIDAD DE TIPOS



ESTRICTAMENTE hablando, una expresión mixta es aquella que mezcla datos de tipos básicos diferentes (números con caracteres, números con datos lógicos, etcétera). Pero en sentido más amplio puede aplicarse también este nombre a las expresiones donde aparecen mezclados datos del mismo tipo básico (numérico) pero de distinto tipo interno (datos enteros con reales, por ejemplo). ¿Qué sucederá en estos casos? ¿Qué acciones toma el traductor (compilador o intérprete) del lenguaje de programación en que escribimos estas expresiones? Es importante saberlo, pues en caso contrario, los resultados pueden ser realmente sorprendentes o imprevisibles.

En un lenguaje como el APL, en el que la representación interna de los datos es invisible para el programador (puesto que externamente sólo existe el tipo numérico, aunque los datos estén de hecho guardados en memoria en la forma más compacta posible) este problema no llega a producirse. También puede prescindirse de él en muchos intérpretes de BASIC, siempre que programemos utilizando sólo una forma de representación interna (números en punto flotante en precisión normal), que es la opción por defecto, pero comienza a tener importancia si encontramos problemas de ocupación de memoria o necesitamos una precisión mayor y hemos de mezclar distintos tipos numéricos en el mismo programa. En PASCAL, donde la mezcla de tipos es usual, el problema de la compatibilidad de tipos es aún más acuciante.

Sin embargo, la cosa no es tan mala como a primera vista pudiera parecer. Todos los intérpretes o compiladores de los lenguajes donde se presenta este tipo de problemas suelen aplicar un corto número de reglas que han sido diseñadas de tal manera que la compatibilidad de tipos sea lo más natural posible. En general, todo dato perteneciente a un tipo más restringido (como el entero) podrá transformarse y operarse sin más con otro dato perteneciente a un tipo más amplio (como uno de los tipos en punto flotante). Los problemas suelen presentarse únicamente cuando deseamos asignarle un valor de tipo más amplio a una variable definida con un tipo más restingido. Pero de ésta ya hemos hablado en el capítulo 10. La conversión de tipo en el lenguaje BASIC se realiza de la misma manera que se explicó en el capítulo 10 para la asignación de valor, a saber:

1. Al convertir de un tipo de precisión más baja a otro de precisión más alta (de entero a real en precisión sencilla o doble; de real en precisión sencilla a real en precisión doble) la conversión se realiza sin dificultades, añadiendo ceros decimales a la derecha del número a convertir. Por ejemplo, el dato de precisión sencilla 2.345678 se convertirá en precisión doble en 2.34567800000000.

NOTA: Debido a que todas las formas de representación interna de los datos

numéricos en los ordenadores tienen cierto error intrínseco, este error se transmitirá también durante las conversiones de tipo. En el caso anterior, el número 2.345678 podría no ser representable exactamente dentro del ordenador. Quizá aparezca (en precisión sencilla) como 2.34567799. En tal caso, al convertirlo a precisión doble pasaría a ser 2.34567799000000. Pero también es posible que este número tampoco sea representable con exactitud en precisión doble. En tal caso, el valor exacto que tendremos después de la conversión podría llegar a ser tan distinto del original como 2.34567799000001.

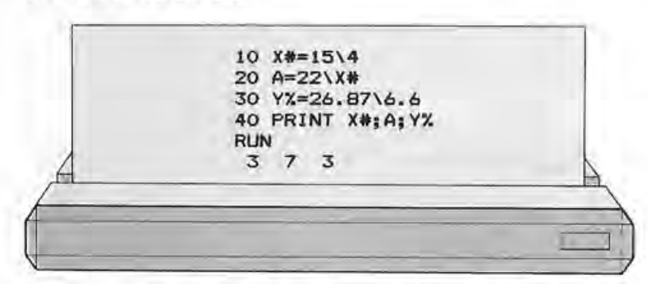
2. Al convertir de un tipo de precisión más alta a otro de precisión más baja, se realiza un redondeo. Por ejemplo, si pasamos a entero el número 2,34, se nos convertirá en 2. En cambio, el número 2.67 se convertirá en 3. Por convenio, en casi todos los intérpretes BASIC, el número 2.5 (que está situado exactamente en el centro del intervalo) se redondea siempre por exceso. Es decir, se convertirá en 3.

Las reglas utilizadas por los intérpretes de BASIC para las expresiones con mezcla de tipos internos son las siguientes:

1. Si la operación a realizar exige un tipo determinado, todos los datos a operar se convertirán a dicho tipo antes de realizar la operación.

2. Si la operación a realizar no exige un tipo determinado, todos los datos a operar se convertirán al tipo más amplio entre los presentes.

Veamos algunos ejemplos de ambos casos. La operación «división entera» (representada por el símbolo \) exige el tipo entero para los datos a los que se aplica. Por tanto, si tratamos de ejecutarla con datos con decimales, éstos se redondearán a enteros antes de que la operación se realice.



Veamos lo que ha ocurrido en este programa: la línea 10 calcula la división entera de dos números enteros: 15 y 4. Todo está, por tanto, en orden. No hay que realizar conversión alguna. El cociente entero de ambos números es 3 y éste es el valor que se le asigna a la variable X#, que está definida (por su nombre) como real en precisión doble. Por tanto, el valor entero 3 debe convertirse durante la asignación a este tipo, más preciso. Supondremos que el dato 3 puede representarse exactamente en la memoria del ordenador. El valor de X# será, por tanto, 3.00000000000000 (o, simplemente, 3).

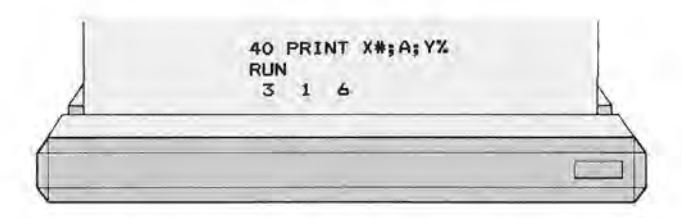
La línea 20 calcula la división entera del entero 22 y de la variable en precisión doble X#. Esta última debe convertirse, por tanto, al tipo entero, redondeando su valor si es necesario. Como éste era 3, no hace falta aplicar redondeo alguno. El resultado será, de nuevo, el entero 3. Ahora hay que calcular el cociente entero de 22 entre 3, que resulta ser 7. Por último, hay que asignar este valor a la variable real en precisión simple A. Supondremos que 7, en precisión simple, se convierte en 7.000000 (o, simplemente, 7).

La línea 30 calcula el cociente entero de dos números con decimales (expresados, por tanto, en precisión simple). Es preciso convertirlos a enteros antes de realizar la operación. El primero (26.87) se redondea a 27. El segundo (6.6) se redondea a 7. Ahora dividimos 27 entre 7, obteniendo un cociente entero igual a 3. Por último, es preciso asignar este resultado a la variable entera Y%. Aquí no hay conversión de tipos y la asignación puede realizarse directamente.

Finalmente, la línea 40 escribe en la pantalla los valores de las tres variables. Podemos comprobar que son iguales a 3, 7 y 3, como acabamos de deducir.

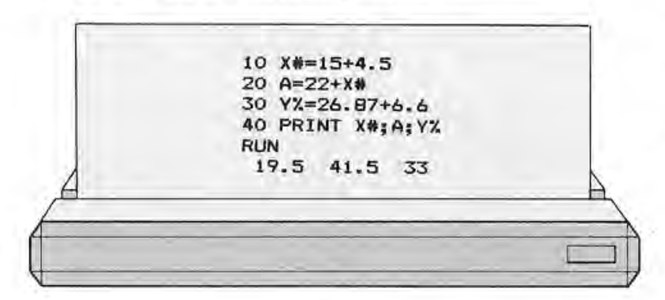
Veamos otro ejemplo. La operación «resto entero» (representada por la palabra reservada MOD) exige también el tipo entero para los datos a los que se aplica. Por tanto, el proceso a seguir será muy parecido al del ejemplo anterior.

```
10 X#=15 MOD 4
20 A=22 MOD X#
30 Y%=26.87 MOD 6.6
```



La explicación de los resultados obtenidos es totalmente equivalente a la del caso de la división entera, por lo que ya no vamos a entrar en detalle.

Veamos ahora un ejemplo de la utilización de operaciones que no exigen un tipo determinado en los datos a los que se aplican. Por ejemplo, la suma.



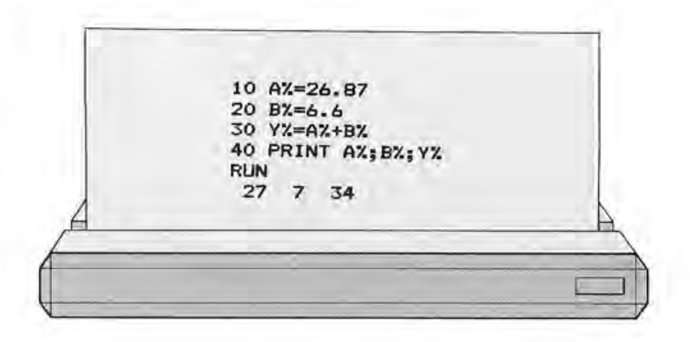
Veamos lo que ha ocurrido en este programa: la línea 10 calcula la suma de un número entero (15) y otro real en precisión simple (4.5). Todos los datos deben convertirse al tipo de dato de más precisión (es decir, a reales en precisión simple) antes de que la operación pueda realizarse. Por tanto, tendremos que sumar 15.00000 y 4.500000, obteniendo el resultado 19.50000. Este es el valor que se le asigna a la variable X#, que es real en precisión doble. Por tanto, el valor real en precisión sencilla 19.5 debe convertirse durante la asignación a este tipo, más preciso. El valor de X# será, por tanto, 19.5000000000000 (o, simplemente, 19.5).

La línea 20 calcula la suma del entero 22 y de la variable en precisión doble X#. Por tanto, la operación va a efectuarse en precisión doble y el número entero deberá convertirse a este tipo (22.0000000000000). El resultado es el número en precisión doble 41.5000000000000. Ahora hay que asignar este valor a la variable real en precisión simple A. Por tanto, habría que convertirlo a un tipo de precisión más baja, redondeándolo. Despreciando los errores de representación en memoria, obtendremos 41.50000 (o, simplemente, 41.5).

La línea 30 calcula la suma de dos números con decimales expresados en precisión simple. La operación se realiza, por tanto, en dicha precisión y no es preciso realizar conversión alguna. El resultado es 33.47. Ahora es preciso asignar este resultado a la variable entera Y%. Por ello es preciso redondear 33.47 al entero más próximo, que es 33.

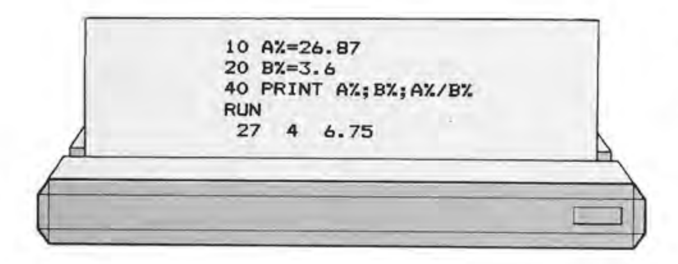
Finalmente, la línea 40 escribe en la pantalla los valores de las tres variables. Podemos comprobar que son iguales a 19.5, 41.5 y 33, como acabamos de deducir.

Obsérvese cómo el resultado habría sido distinto si la conversión a entero se hubiera realizado antes de efectuar la operación. En tal caso, 26.87 se habría convertido en 27 y 6.6 en 7, con lo que su suma sería igual a 34. Si quisiéramos obtener dicho resultado, podríamos lograrlo con las siguientes instrucciones BASIC:



En este programa, los dos valores reales (26.87 y 6.6) se asignan primero a las dos variables enteras A% y B%, respectivamente. Se produce, por tanto, una conversión de tipo que redondea ambas a los valores 27 y 7. En la línea 30, las dos variables A% y B% (que ya eran enteras) se suman, y se asigna el resultado a la variable Y%, también entera. Así, pues, la operación se realiza directamente en el tipo entero y no es preciso realizar conversión alguna. La suma de 27 y 7 da, como es natural, el resultado 34, que es el nuevo valor de la variable Y%.

El tipo del resultado será, en general, el mismo que el de los datos con los que se opera, pero no siempre. La división con decimales, por ejemplo (representada con el símbolo /) produce siempre un resultado en punto flotante, aunque los dos datos a los que se aplique sean enteros. Por ejemplo:

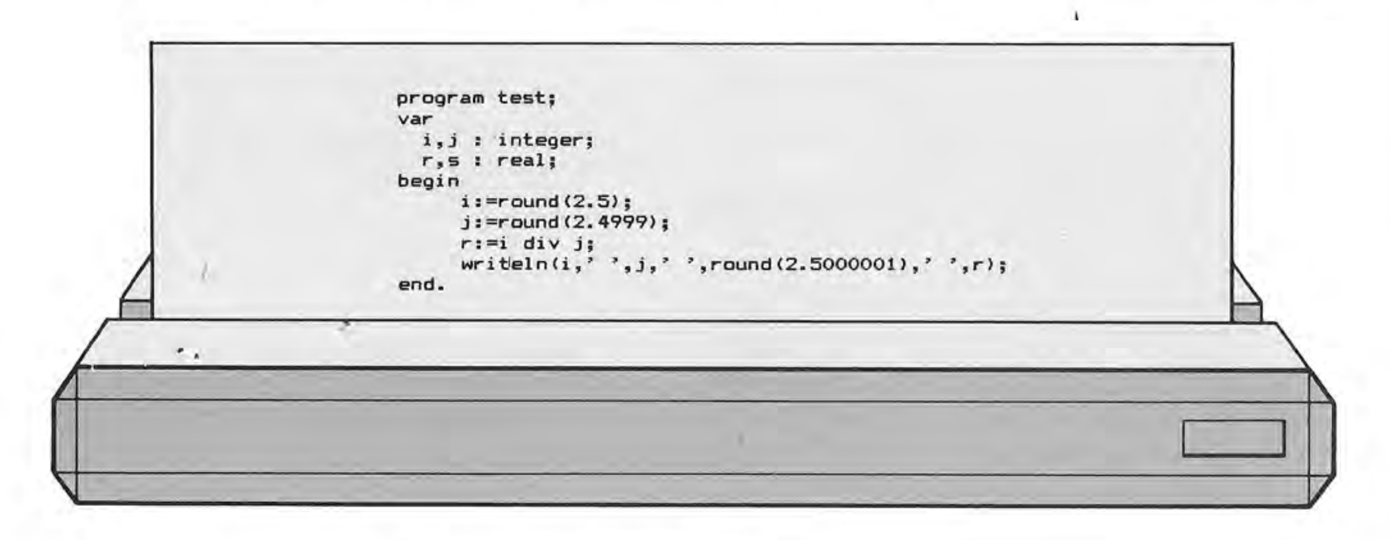


```
10 A%=26.87
20 B%=3.6
40 PRINT A%;B%;A%/B%
RUN
 27  4  6.75
```

En PASCAL, el problema de las conversiones es menos acuciante, pues sólo hay dos tipos numéricos en lugar de tres: enteros y reales. La conversión de entero a real no tiene problema, y se lleva a efecto automáticamente siempre que es necesaria. Para la conversión de real a entero, ya vimos en el capítulo 10 que los compiladores de PASCAL reconocen dos funciones básicas: TRUNC, que trunca siempre por defecto los valores reales a los que se aplica, y ROUND, que los redondea al entero más próximo. En general, si una operación exige los datos en el tipo entero (como en el caso de DIV y MOD), no podrá aplicarse nunca a datos del tipo real, pues el compilador generará un mensaje de error. Es decir, al revés que en el BASIC, en PASCAL no hay redondeo automático. Otra diferencia es que ROUND (2.5) es igual a 2. Es decir, el redondeo se hace en este caso por defecto.

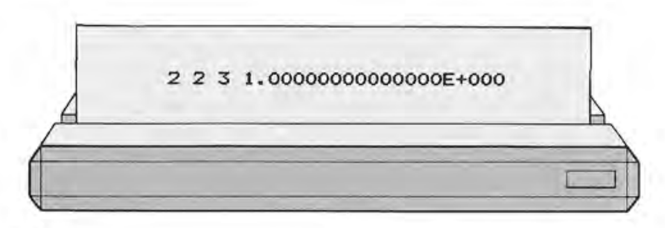
Las operaciones aritméticas de PASCAL se realizan normalmente, cuando pueden actuar con datos de varios tipos y se aplican a más de un dato, en el tipo del que tiene más precisión. Si los dos datos son enteros o reales, la operación se realizará normalmente sin aplicarles conversión alguna. Si uno de ellos es entero y el otro real, el entero se convertirá automáticamente a real antes de realizar la operación. En cuanto al resultado, pertenecerá normalmente al tipo en que se realizó la operación. La excepción es, también en PASCAL, la división de números reales (representada por el símbolo /), que convierte siempre sus datos al tipo real, aunque los dos fueran enteros inicialmente.

Si deseamos realizar las operaciones DIV y MOD con datos reales, será preciso convertir éstos primero al tipo entero mediante las funciones básicas TRUNC o ROUND. Por ejemplo:



```
program test;
var
  i,j : integer;
  r,s : real;
begin
     i:=round(2.5);
     j:=round(2.4999);
     r:=i div j;
     writeln(i,' ',j,' ',round(2.5000001),' ',r);
end.
```

que al ser ejecutado da el siguiente resultado:



```
2 2 3 1.00000000000000E+000
```

APLICACIONES

Hojas de cálculo. Multiplán



EL Multiplán es una de las hojas de cálculo de más difusión en el mercado actual. Influyen dos cosas de forma determinante en ello.

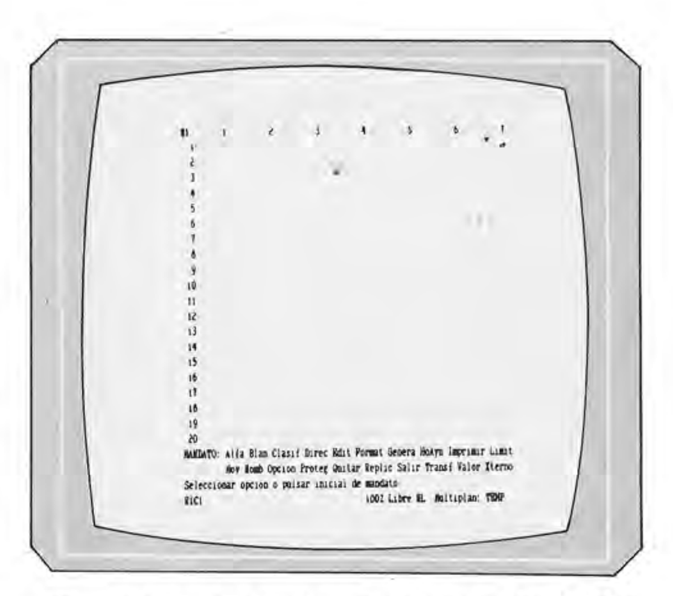
— Las versiones existentes para los distintos microordenadores, que la convierten en una hoja de cálculo casi universal.

— La gran facilidad para traspasar datos a otros programas permitiendo así combinar la versatilidad del Multiplán como hoja electrónica con las posibilidades ofrecidas por programas de gráficos, procesadores de textos, bases de datos, etc.

Para acceder al programa basta introducir las letras MP seguidas del número de columnas deseado para el monitor: 40 u 80. En general se trabaja con MP80, ya que no todos los monitores admiten MP40.

Aparece en el monitor la pantalla del Multiplán que consta de:

— Retícula de filas y columnas.
— Menú de comandos.
— Línea de explicación del comando.
— Celda activa (sobre la que se encuentra el cursor o se está trabajando).
— Volumen de la hoja de cálculo que se encuentra sin ocupar.
— Nombre de la hoja de cálculo: Por defecto toma el nombre de Temp, que significa que se trata de una hoja temporal.



El menú de comandos hace de Multiplán un programa de fácil aprendizaje; prácticamente sin manual usted puede llegar a conseguir realizar una hoja de trabajo, simplemente debe dejarse guiar por el menú.

Movimiento por la hoja

Para desplazarse por las celdas utilice las flechas de movimiento, éstas le permitirán pasar a la celda contigua. Si desea un movimiento más rápido, utilice las teclas (Pg Up) y (Pg Dn), según quiera ir una página arriba o abajo. Para desplazarse por el menú de comandos utilice las teclas de función F9 y F10. Si lo prefiere, puede acceder al comando directamente pulsando la inicial del mismo.

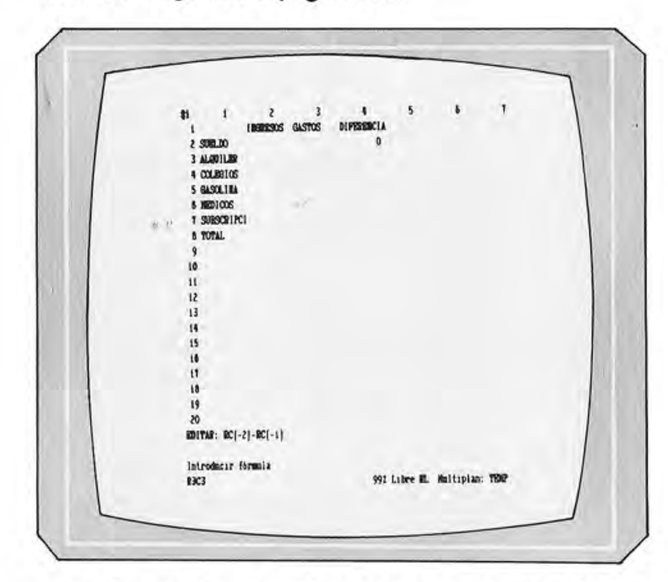
Dentro de los menús el desplazamiento se realiza con los tabuladores.



Comandos

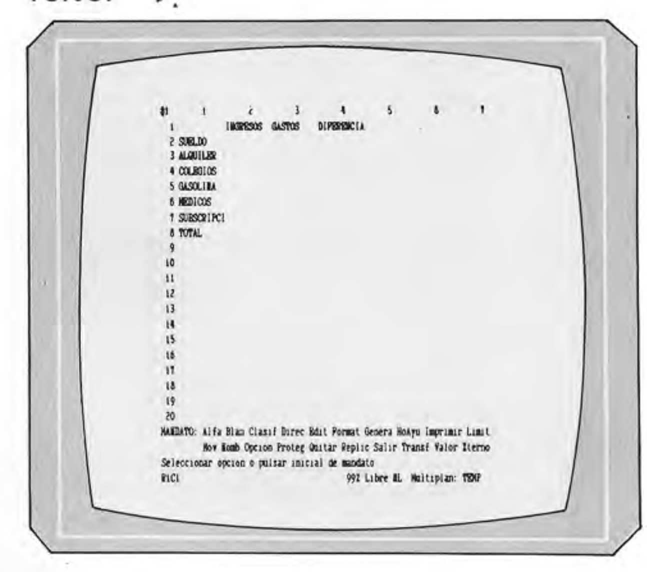
Vamos a ir viendo, uno a uno, los comandos que aparecen en el menú:

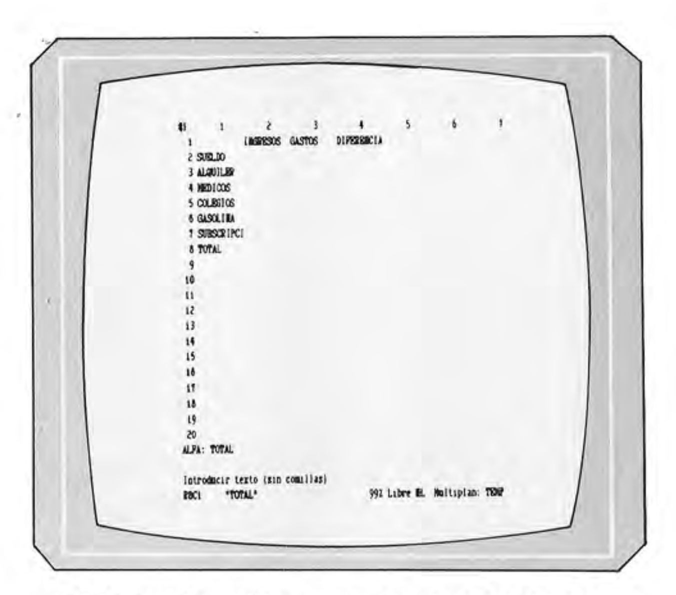
ALFA. Permite crear un literal en una celda. Creamos, por ejemplo, literales en R1C2, R1C3, y R1C4 para indicar ingresos, gastos y diferencia entre ambos, y en la columna 1 introducimos los motivos de dichos ingresos y gastos.



BLAN. Rellena de blancos la celda o rango de celdas indicadas.

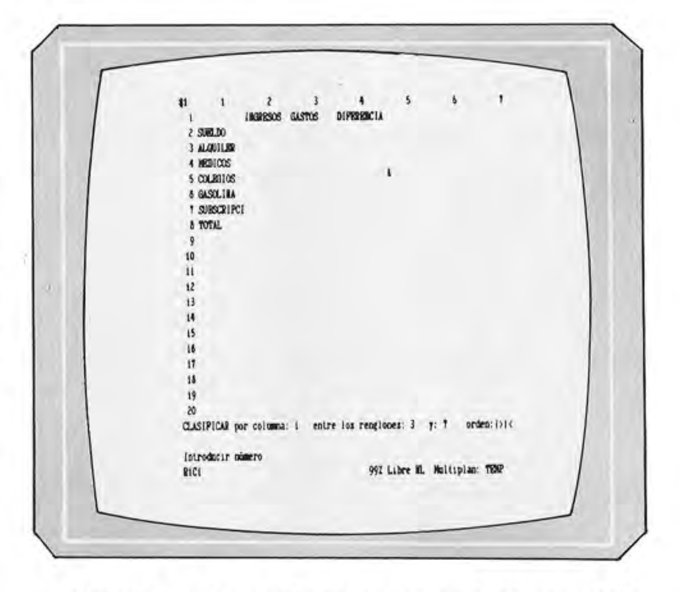
CLASIF. Con este comando se puede ordenar de mayor a menor y por orden alfabético, bien por filas o por columnas. Si la fila o columna (permite, también, sólo una parte de la misma) es numérica, ordenará de mayor a menor o de menor a mayor el grupo de celdas especificadas. Si se trata de celdas alfanuméricas (literales), la ordenación creciente, de mayor a menor, supone una ordenación alfabética empezando por la A, y la decreciente será un orden alfabético inverso.





DIREC. Se utiliza para direccionar un nombre, un renglón o una columna.

EDIT. Permite escribir fórmulas en una celda y editar el contenido para modificarlo.

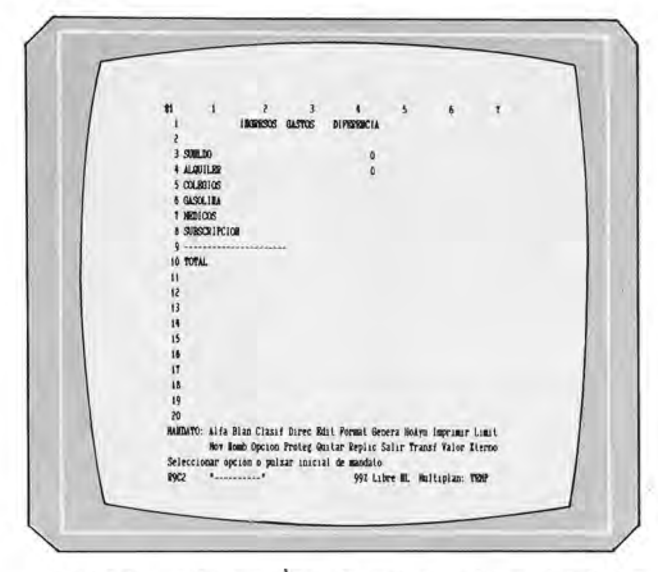


FORMAT. Con diferentes opciones, este comando nos permite determinar el formato que deseemos para una celda o rango de celdas. Podremos variar atributos tales como la longitud, si el contenido se pega a la derecha, centro o izquierda, tipo de decimales y número, caracteres especiales, si deben aparecer las fórmulas en pantalla o si están permitidos los puntos o no.

Reformemos, por ejemplo, la longitud de la columna 1 para que el literal suscripción quepa entero en la celda.

GENERA. Este comando genera renglones o columnas en la posición indicada.

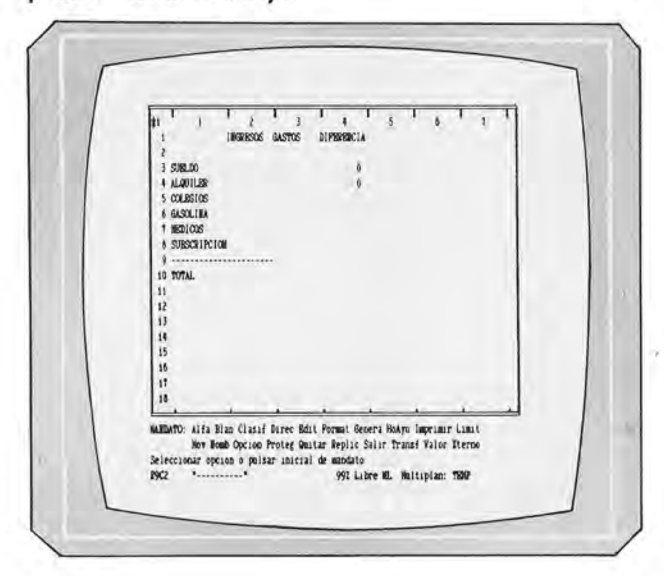
Continuando con el ejemplo, generamos una línea encima de la 2 y otra entre la 7 y la 8 para separar un poco las zonas. En la nueva fila 8, mediante el comando ALFA, rellenamos con guiones para subrayar.



HOAYU. Nos permite acceder al completo help que posee el Multiplán.

IMPRIMIR. Con varias opciones adicionales es el comando que permite volcar la hoja de trabajo o una zona de ella en la impresora. Es muy importante la opción MARGENES, que mediante un comando adecuado de compresión permite aumentar el número de columnas enviadas a la impresora.

LIMIT. Es el comando que permite dividir la hoja en ventanas y crear limitadores. Ejemplos de ventanas se verán en el siguiente fascículo. Creemos un limitador para nuestra hoja:



MOV. Con este comando se puede cambiar de lugar una columna o una fila, desapareciendo de su antigua posición.

NOMB. Permite dar nombre a una celda o grupo de celdas, de forma que luego nos podemos referir a ella/s por su nombre (incluso en las fórmulas).

OPCION. Permite anular la recalculación automática de la hoja de cálculo, es muy útil cuando el volumen de la misma es elevado, ya que cualquier modificación en esas condiciones supondría tiempo de recalculación. Se recomienda, por tanto, desconectar RECALCULAR y conectarlo cuando todos los datos estén introducidos.

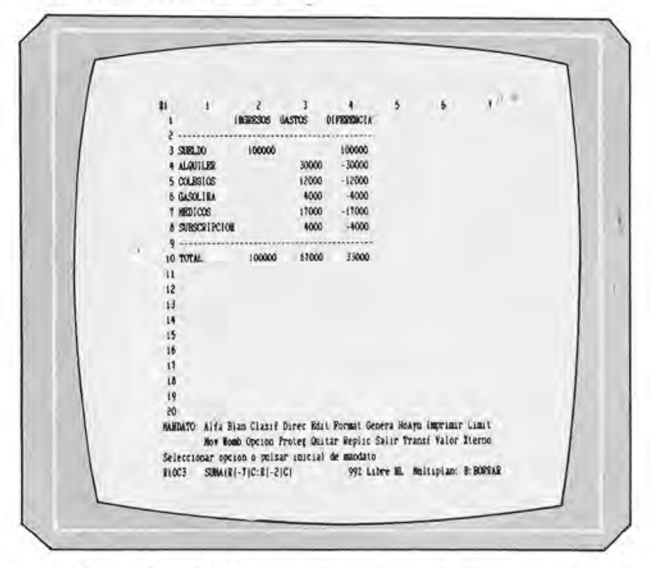
En este comando se encuentra también la posibilidad de realizar iteraciones y poner alarmas para situaciones indeseadas.

PROTEG. Se utiliza para proteger las celdas. Una celda protegida no puede ser modificada. Se recomienda utilizarlo con las fórmulas, una vez comprobado y verificado su funcionamiento, para evitar que por un error se pierdan.

QUITAR. Su misión es eliminar la celda o rango de celdas especificado, es por ello un comando algo peligroso.

REPLICAR. Permite copiar el contenido de una celda/s a otra/s y copiar en vertical u horizontal el contenido de una celda. Es muy útil para las fórmulas.

Como ejemplo vamos a copiar las líneas de guiones para completarlas y las fórmulas de diferencia. Introducimos las fórmulas de total.



SALIR. Se utiliza para abandonar el Multiplán. Antes de salir pregunta si está se-

guro con la intención de que si hay información nueva sin almacenar ésta no se pierda.

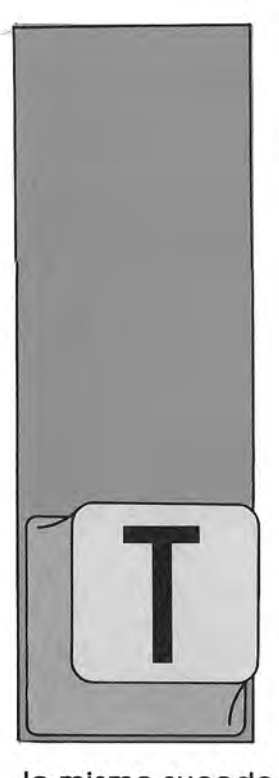
TRANSF. Con diferentes opciones, nos ofrece la posibilidad de almacenar la hoja de trabajo en el disco, cargar una hoja ya existente en él o almacenarlo como un fichero estándar.

VALOR. Se utiliza para dar valores numéricos a las celdas. Si no damos ningún comando y directamente escribimos un número, Multiplán entiende que queremos dar valor a la celda activa asumiendo el comando.

XTERNO. Se utiliza para transferir datos de y a otros programas.

Queda terminada la exposición de los comandos, cuyo funcionamiento, como puede verse, es bastante sencillo. Compruebe su funcionamiento e introduzca ejemplos, es el mejor modo de familiarizarse con un programa. Multiplán no sólo es una hoja de cálculo de fácil manejo, es, además, una hoja potente y versátil, con posibilidades en cualquier campo.

PASCAL



Tipos estructurados de datos

TODOS los tipos de datos vistos hasta el momento sirven para manejar datos simples: una variable de tipo INTEGER sólo puede tomar un valor en un momento dado, y lo mismo sucede con los tipos CHAR, DíaDeLaSemana y los demás que conocemos.

Sin embargo, nos podría interesar guardar en una única variable el peso, la edad, la estatura y el nombre de una persona, o guardar en una sola variable las notas del examen de Química de todo un grupo de alumnos, etc.

En PASCAL se pueden definir tipos de datos que a su vez se componen de otros tipos, para así poder tratar con varios valores a la vez; son los llamados TIPOS ESTRUCTURADOS. Vamos a ver a continuación el más común de ellos, el tipo ARRAY o tabla.



El tipo ARRAY

Supongamos por un momento que queremos obtener la media de las notas de un examen y posteriormente obtener la nota más alta; primero deberíamos sumarlas todas y dividir el resultado por el número de notas. Haríamos:

```
program Notas;

var
 Nota1, Nota2, Nota3,
 Suma, Media, Maxima : real;

begin
 writeln ('Introduzca las notas:');
 readln (Nota1);
 readln (Nota2);
 readln (Nota3);

    (* Calculamos la suma: *)
 Suma:= 0.0;
 Suma:= Suma + Nota1;
 Suma:= Suma + Nota2;
 Suma:= Suma + Nota3;
    (* Obtenemos la media: *)
 Media:= Suma / 3.0;
 writeln ('La nota media es',Media:6:2);

  (* Ahora obtenemos la nota máxima *)
 Maxima:= 0.0;
 IF Nota1 > Maxima then Maxima:= Nota1;
 IF Nota2 > Maxima then Maxima:= Nota2;
 IF Nota3 > Maxima then Maxima:= Nota3;
 writeln ('La nota más alta es',Maxima:6:2);
end.
```

Es fácil imaginar cómo sería este programa para calcular la media, no de tres, sino de cien notas. Si se pudiesen guardar todas en memoria como un grupo y no cada una con su propia variable el programa sería mucho más sencillo.

Las variables de tipo ARRAY permiten guardar en una sola de ellas muchos datos del mismo tipo. Es como si tuviésemos una tabla con muchos valores registrados y para escoger uno de ellos dijéramos: «dame el primer valor de la tabla», o el quinto, o el tercero, etc.

Para utilizar un elemento en concreto de la tabla se necesita, como es lógico, indicar cuál de ellos es, y esto se hace por medio de lo que se denomina INDICE.

Este índice podría ser un valor de tipo INTEGER, con lo que podríamos decir algo como «Guarda esto en el elemento número 5 de la tabla» o «Presenta en pantalla el elemento número 2».

Por tanto, al definir una variable del tipo ARRAY hay que indicar dos cosas:

— Entre qué valores puede estar comprendido el índice; en otras palabras, el subrango al que pertenece éste.

— De qué tipo de elementos consta la tabla; podría ser una tabla de números enteros o reales, de caracteres, de variables del tipo DiaDeLaSemana...

Para ello se escribe en primer lugar la palabra reservada ARRAY seguida del subrango al que corresponda el índice puesto entre corchetes; tras ello se escribe la palabra reservada OF seguida del tipo correspondiente a los elementos de la tabla. Por ejemplo:

```
var Notas: array [1..100] of integer;
```

así, la variable Notas sería una tabla de números de tipo INTEGER; los elementos de la tabla se escogerían indicándolo con un índice que podría estar comprendido entre 1 y 100. De esta manera, tendríamos el elemento número 1 de la tabla, el 2, etc., hasta el 100, o sea, 100 elementos en total.

Como hemos visto en otras ocasiones, puede ser más conveniente no definir la variable sobre la marcha y escribir:

```
type
   Notas_t = array [1..100] of integer;
var
   NotasClase1, NotasClase2: Notas_t;
```

o, incluso:

```
type
   Indice_t = 1..100;
   Notas_t = array [Indice_t] of inte-
ger;
var
   NotasClase1, NotasClase2: Notas_t;
```

Para indicar el elemento de la tabla que se desea utilizar se escribe el nombre de ésta seguido de una constante, variable o expresión que proporcione el valor del índice, puesta entre corchetes:

```
(* Escribe el doble del      *)
(* elemento 1 de la tabla: *)

write (Notas[1] * 2);

(* Guarda 7 en el elemento *)
(* 100 de la tabla:        *)

Notas [50 * 2]: 7;
```

Los elementos así especificados son del tipo constitutivo del ARRAY; en el ejemplo serían, pues, del tipo INTEGER y por tanto se podrían utilizar exactamente igual que cualquier otra variable INTEGER.

Si nuestro ordenador no dispone de los corchetes en su juego de caracteres, se pueden utilizar en su lugar las parejas (. y .):

```
Notas (.5.):= 3;
```

No se pueden comparar variables de tipo ARRAY entre sí, pues no tendría sentido, ni hacer operaciones con ellas tomadas en su conjunto. La única opera-

ción posible con todos los elementos de un ARRAY a la vez es la asignación:

NotasClase1:= NotasClase2;

esto guardaría todos los valores de la tabla NotasClase2 en la tabla NotasClase1, es decir, el elemento 1 de NotasClase2 en el 1 de NotasClase1, el 2 en el 2, etc.

Hay que hacer notar que el número de elementos de la tabla queda definido por el subrango del índice, que es fijo. Si al escribir el programa especificamos, por ejemplo, 21..30, la tabla tendrá 10 elementos y esto no podrá ser cambiado durante la ejecución del programa.

Si quisiéramos preparar el programa Notas para funcionar con una tabla en lugar de con una variable distinta para cada nota, pondríamos en principio:

```
...
writeln ('Introduzca las notas:');
   readln (Notas[1]);
   readln (Notas[2]);
   readln (Notas[3]);
   ...
```

con lo que, ciertamente, se gana poco. Sin embargo, como el índice puede ser una variable, el paquete de instrucciones READLN se puede sustituir por un bucle FOR en que la variable de control se utilice como índice, y lo mismo sucede para las demás instrucciones que se repiten.

Estamos ya en condiciones de hacer un programa de medias mejorado;

```
program NotasMejor;

const
  Tope= 100;  (* Como mucho podrá haber 100 notas *)

type
  Indice_t = 1..Tope;
  Notas_t  = array [Indice_t] of real;
  (* Las notas pueden tener decimales, de ahí el tipo REAL *)

var
  Notas               : Notas_t;
  Suma, Media, Maxima : real;
  Indice, Total       : Indice_t;

begin
 writeln ('El máximo número de notas es ',Tope);
 write ('Cuantas notas ? '); readln (Total);

 (* Vamos a leer las notas tecleadas *)
 for Indice:= 1 to Total do
   begin
    write ('Nota número ',Indice,' = ');
    readln (Notas[Indice])
     (*  Como un REAL se puede leer de teclado,  *)
     (*  los elementos de NOTAS también.         *)
   end;

 Suma:= 0.0;
    (* Vamos acumulando notas en Suma: *)
 for Indice:= 1 to Total do Suma:= Suma + Notas [Indice];
    (* Ahora calculamos la media *)
 Media:= Suma / Total;
 writeln ('La nota media es',Media:6:2);

 (* Ahora obtenemos la nota máxima *)
 Maxima:= 0.0;
 for Indice:= 1 to Total do
   if Notas [Indice] > Maxima then Maxima(*por ahora*):= Notas [Indice];
  writeln ('La nota más alta es',Maxima:6:2);
end.
```

Como se ve, en lugar de utilizar una instrucción para cada nota se han utilizado bucles FOR, pues las instrucciones se repiten un número fijo de veces. Además, se ha descrito el subrango del índice previamente para así no tener que repetirlo con Indice y Total. Nótese también dónde se ha puesto uno de los comentarios: recordemos que se pueden poner comentarios en cualquier lugar en que pueda ir un espacio en blanco.

Para terminar por ahora, vamos a modificar el programa del Master Mind que hicimos en el número anterior utilizando tablas de cuatro enteros para guardar las combinaciones y aprovechando las ventajas que esto nos brinda para hacer el análisis de una manera más elegante:

```
program  MasterMind;

 type
  Combi_t = array [1..4] of integer;

 var
  Aleatorio    : real;

  Secreta,      (* Para guardar la clave secreta *)
  Intento:      (* Para guardar cada intento     *)
                 Combi_t;
  Muertos,
  Heridos,
  Limite,
  Indice       : integer;
  Letra        : char;

(*---------------------------------------------------------*)
procedure ArrancaAzar;
(* Pide el primer número de la serie aleatoria *)

  var Ok: boolean;
 begin
  writeln ('Teclee un número entre 0 y 1, ambos exclusive.');
  repeat
   readln (Aleatorio);
   Ok:= (0.0 < Aleatorio) and (Aleatorio < 1.0);
   if not Ok then writeln ('No vale. Repita.')
  until Ok
 end;

(*---------------------------------------------------------*)
function Azar (Inf,Sup: integer): integer;
(*   proporciona un número entero aleatorio    *
 *   entre Inf y Sup, ambos inclusive          *)

  var A: real;
 begin
  Aleatorio:= frac (Aleatorio * 997);            (* ¡OJO! *)
  Azar:= Inf + trunc (Aleatorio * (Sup - Inf + 1))
 end;

(*---------------------------------------------------------*)
procedure Analizar;
 (*  Compara Secreta con Intento y da          *
  *  el valor adecuado a Muertos y Heridos. *)
  var
   Copia: Combi_t;
   I, J : integer;
 begin
  (* Saca una copia de la clave secreta: *)
  Copia:= Secreta;

  Muertos:= 0;
  for I:= 1 to 4 do
    if Intento [I] = Copia [I] then
      begin
       Muertos:= Muertos + 1;
       Intento [I]:= -1;
       Copia [I]  := -2
      end;

  Heridos:= 0;
  for I:= 1 to 4 do     (* mira las 4 cifras de Intento: *)
    (*------------------------------*)
    for J:= 1 to 4 do  (* compara Intento [I] con Copia *)
       if Intento [I] = Copia [J] then
          begin
            Heridos:= Heridos + 1;
            Intento [I]:= -1;
```

```
            Copia [J]  := -2
          end
      (*--------------------------------*)
  end;

 (*-----------------------------------------------------------*)
 begin
  ArrancaAzar;
  write ('Números del 1 al...');
  readln (Limite);
  (*-----------------------------------------*)
  repeat              (* repetir partidas *)
    (* Primero, inventarse una clave: *)
    for Indice:= 1 to 4 do Secreta [Indice]:= Azar (1, Limite);

    ClrScr; (* o PAGE, para borrar la pantalla *)
    writeln ('Tras cada cifra, pulse Intro.');
    writeln;
    (*-----------------------------------------*)
    repeat  (* Pedir y analizar combinaciones: *)
      write ('Clave:  ');
      for Indice:= 1 to 4 do read (Intento [Indice]);

      Analizar;

      writeln ('  m=',Muertos,'  h=',Heridos)
    until Muertos = 4;
    (*-----------------------------------------*)
    writeln;
    write ('¿ Desea otra partida ? (S/N)  ');
    readln (Letra)

  until (Letra = 'N') or (Letra = 'n');
  (*-----------------------------------------*)
  writeln ('Adiós.')

 end.
```

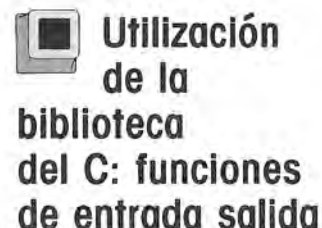
Compare el lector este programa con la versión anterior. Al hacerse la búsqueda de heridos utilizando un bucle FOR dentro de otro, se compara Copia(1) con Intento(1), Copia(2) con Intento(2), etc., lo cual es incorrecto; no obstante, si se diera alguna igualdad entre ellos, ya se habría detectado al mirarse los muertos, por lo que estarían «tachados» y nunca podrían ser iguales al mirarse los heridos, así que no pasa nada.

Como ejercicio, pruebe el lector a hacer variable el número de cifras con que se juega en lugar de que sea siempre cuatro.

OTROS LENGUAJES

LENGUAJE C: FUNCIONES





Utilización de la biblioteca del C: funciones de entrada salida

SI deseamos utilizar una función de la biblioteca en nuestro programa fuente, deberemos emplear la línea «#include <stdio.h>»

Include <stdio.h>

se emplea para que todo archivo fuente pueda utilizar las funciones de la biblioteca. El fichero <studi.h> contiene macros y variables utilizadas por la biblioteca.

Función para abrir ficheros

La función utilizada para la apertura de ficheros es «fopen()»:

FILE *fopen()

La función «fopen()» devolverá un valor «NULL» en caso de que le sea imposible abrir el fichero.

Más funciones de E/S

Si tenemos que leer un carácter del dispositivo de entrada lo haremos a través de la función:

getc()

Si deseamos enviar un carácter a un fichero apuntado por el puntero «puntarch», deberemos emplear «putc()» de la siguiente forma:

putch (ch,puntarch)

Tanto «get()» como «put()» se utilizan sólo para ficheros de texto.

Para cerrar un fichero utilizaremos la función:

fclose()

Para cerrar el fichero apuntado por «puntarch» escribiremos:

fclose (puntarch);

Ficheros estándar

Los tres ficheros estándar en C son:

«stdin», «stdout», «stderr»

Los tres ficheros son abiertos antes de «llamarse» a la función «main()». Estos tres archivos se denominan «entrada estándar», «salida estándar» y «salida estándar errores».

Salidas y entradas con formatos

Las funciones básicas para la salida y entrada con formatos son:

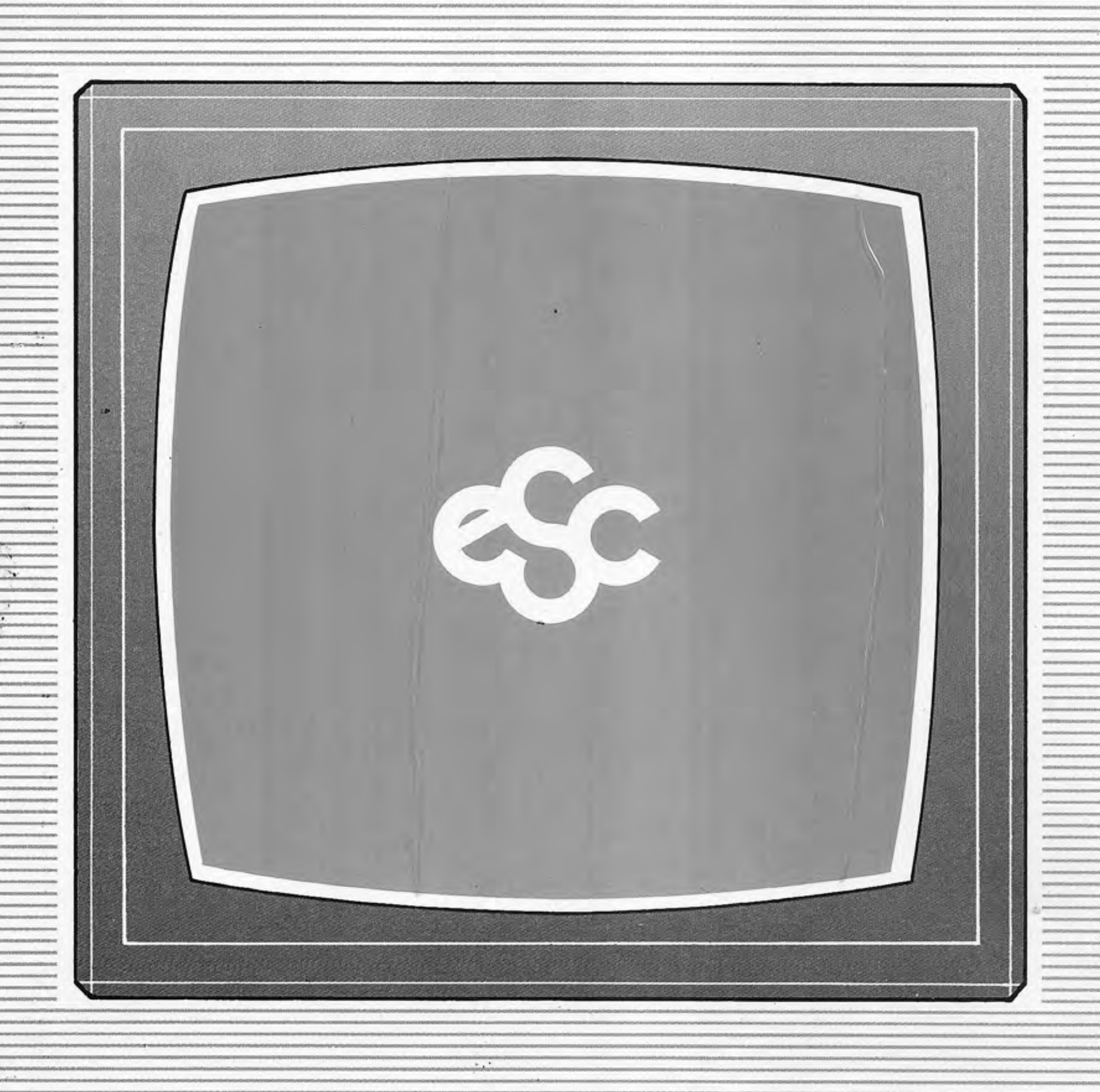
printf() y scanf()

La función «printf()» nos permite convertir, dar formato e imprimir en el dispositivo de salida estándar una serie de argumentos definidos en la función, bajo el control del «parámetro de control».

La sintaxis utilizada será:

printf (argumento de control, argumento 1, argumento 2,...)

El parámetro de control establece dos tipos de salida: por una parte, los caracteres ordinarios que simplemente se copian a la salida estándar y, por otra, las «especificaciones de conversión» que origina la impresión de los argumentos.



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼